Commission on Crystallographic Computing
International Union of Crystallography
http://www.iucr.org/iucr-top/comm/ccom/
Newsletter No. 7, November 2006
(6th June 2008: updated least-squares article)
This issue:
"Understanding Crystal Structures"
http://www.iucr.org/iucr-top/comm/ccom/newsletters/

Table of Contents

(This issue's editors: Simon Parsons and Lachlan Cranswick)

(Warning – unless you want to kill 166 pages worth of forest – DO NOT press the "print" button. For hardcopies – you may like to only print out the articles of personal interest.)

Chairman: Professor Dr. Anthony L. Spek
Director of National Single Crystal Service Facility,
Utrecht University,
H.R. Kruytgebouw, N-801,
Padualaan 8, 3584 CH Utrecht,
the Netherlands.
Tel: +31-30-2532538
Fax: +31-30-2533940
E-mail: a.l.spek@chem.uu.nl
WWW: http://www.cryst.chem.uu.nl/spea.html
WWW: http://www.cryst.chem.uu.nl/platon/

Lachlan M. D. Cranswick
Canadian Neutron Beam Centre (CNBC),
National Research Council of Canada (NRC),
Building 459, Station 18, Chalk River Laboratories,
Chalk River, Ontario, Canada, K0J 1J0
Tel: (613) 584-8811 ext: 3719
Fax: (613) 584-4040
E-mail: lachlan.cranswick@nrc.gc.ca
WWW: http://neutron.nrc-cnrc.gc.ca/peep_e.html#cranswick

Dr Ralf W. Grosse-Kunstleve
Lawrence Berkeley National Laboratory
One Cyclotron Road, BLDG 64R0121,
Berkeley, California, 94720-8118, USA.
Tel: (510) 486-5929
Fax: (510) 486-5909
E-mail: RWGrosse-Kunstleve@lbl.gov
WWW: http://cctbx.sourceforge.net/
WWW: http://www.phenix-online.org/
WWW: http://cci.lbl.gov/~rwgk/

Professor Alessandro Gualtieri
Università di Modena e Reggio Emilia,
Dipartimento di Scienze della Terra,
Via S.Eufemia, 19,
41100 Modena, Italy
Tel: +39-059-2055810
Fax: +39-059-2055887
E-mail: alex@unimore.it
WWW: http://www.terra.unimo.it/en/personaledettaglio.php?user=alex

Professor Luhua Lai
Institute of Physical Chemistry,
Peking University,
Beijing 100871, China.
Fax: +86-10-62751725.
E-mail: lhlai@pku.edu.cn
WWW: http://mdl.ipc.pku.edu.cn/

Dr Airlie McCoy
Structural Medicine,
Cambridge Institute for Medical Research (CIMR)
Wellcome Trust/MRC Building,
Addenbrooke's Hospital,
Hills Road, Cambridge CB2 2XY, UK
Tel: +44 (0) 1223 217124
Fax: +44 (0) 1223 217017
E-mail: ajm201@cam.ac.uk
WWW: http://www.haem.cam.ac.uk/
WWW: http://www-structmed.cimr.cam.ac.uk/

Professor Atsushi Nakagawa
Research Center for Structural and Functional Proteomics,
Institute for Protein Research, Osaka University,
3-2 Yamadaoka, Suita, Osaka, 565-0871, Japan
Tel: +81-(0)6-6879-4313
Fax: +81-(0)6-6879-4313
E-mail: atsushi@protein.osaka-u.ac.jp
WWW: http://www.protein.osaka-u.ac.jp/rcsfp/supracryst/

Dr. Simon Parsons
School of Chemistry
Joseph Black Building,
West Mains Road,
Edinburgh, Scotland, EH9 3JJ, UK
Tel: +44 131 650 5804
Fax: +44 131 650 4743
E-mail: s.parsons@ed.ac.uk
WWW: http://www.crystal.chem.ed.ac.uk/

Dr Harry Powell
MRC Laboratory of Molecular Biology,
Hills Road, Cambridge, CB2 2QH, UK.
Tel: +44 (0) 1223 248011
Fax: +44 (0) 1223 213556
E-mail: harry@mrc-lmb.cam.ac.uk
WWW: http://www.mrc-lmb.cam.ac.uk/harry/

Consultants

Professor I. David Brown
Brockhouse Institute for Materials Research,
McMaster University,
Hamilton, Ontario, Canada
Tel: 1-(905)-525-9140 ext 24710
Fax: 1-(905)-521-2773
E-mail: idbrown@mcmaster.ca
WWW:
http://www.physics.mcmaster.ca/display.php4?page=sw://lists/Minibio_2004.php4?ID=4

Professor Eleanor Dodson
York Structural Biology Laboratory,
Department Of Chemistry,
University of York, Heslington, York, UK, YO10 5YW
Tel: +44 (1904) 328253
Fax: +44 1904 328266
E-mail: e.dodson@ysbl.york.ac.uk
WWW: http://www.ysbl.york.ac.uk/people/6.htm

Dr David Watkin
Chemical Crystallography,
Oxford University,
9 Parks Road,
Oxford, OX1 3PD, UK.
Tel: +44 (0) 1865 272600
Fax: +44 (0) 1865 272699
E-mail: david.watkin@chemistry.oxford.ac.uk
WWW: http://www.chem.ox.ac.uk/researchguide/djwatkin.html

Septième atelier de diffraction sur poudre

Seventh Canadian Powder Diffraction Workshop

Université du Québec à Trois-Rivières

Du mercredi 16 au vendredi 18 mai 2007 - Wednesday 16th to Friday 18th of May 2007

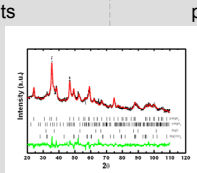
Université du Québec à Trois-Rivières, Trois-Rivières, Québec

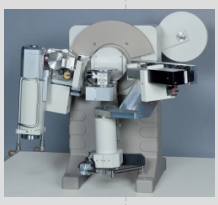
http://www.cins.ca/cpdw/

(L'atelier se déroulera en anglais / Workshop will be in english)

Lecture Titles Include:

- Introduction to Powder Diffraction and Powder Diffraction Hardware
- Introduction to the basics of crystallography
- Sample preparation, data collection and phase identification using powder X-ray diffraction
- Introduction to Powder Profile Refinement
- Synchrotron and Neutron Experiments
- Freely available powder diffraction software
- Profile Refinement with GSAS
- Beyond the Bragg peaks or why do we care about total scattering?
- What to do with your PDF: Modeling of disordered structures





Chair :

- Prof. Jacques Huot *(UQTR)*

Speakers :

- *Dr Robert Von Dreele (ANL, USA)*
- *Dr Angus Wilkinson (Georgia Tech, USA)*
- *Dr Thomas Proffen (Los Alamos, USA)*
- *Dr Ian Swainson and Lachlan Cranswick (NRC-CNRC, Canada)*

À qui s'adresse l'atelier ?

Tous ceux qui s'intéressent ou qui utilisent la diffraction X et la diffraction neutronique sur poudre. Ouvert aux étudiants gradués, techniciens et chercheurs qui utilisent la diffraction X sur poudre et qui désirent se familiariser avec l'analyse Rietveld.

Who should attend ?

Anyone interested in or currently using powder X-ray or neutron diffraction. Open to graduate students, technicians and researchers who use X-ray and neutron diffraction and who would like to learn Rietveld analysis.

Détails / Details

Coûts d'inscription / Registration costs

Universitaire / University based: 200 $ CAN

Régulier / Régular : 450 $ CAN

(Incluant les dîners et les taxes / Including lunchs and taxes)

Hébergement / Housing

Forfait 2 nuits sur le campus: 90$ CAN

2 nights on campus package: 90$ CAN

Date limite d'inscription / Registration deadline :

13 avril 2007 / April 13, 2007

Plus d'information sur le site web :
Check out the following web site :
http://www.cins.ca/cpdw/

Hydro Québec

NRC-CNRC

Canadian Institute for Neutron Scattering

Multipurpose crystallochemical analysis with the program package TOPOS

Vladislav A. Blatov
Samara State University, Ac.Pavlov Street. 1, Samara 443011, Russia. E-mail: blatov@ssu.samara.ru ; WWW: http://www.topos.ssu.samara.ru/blatov.html ; TOPOS website: http://www.topos.ssu.samara.ru/

Abbreviation list

CIF	Crystallographic Information File
CN	Coordination number
CS	Coordination sequence
CSD	Cambridge Structural Database
DBMS	Database Management System
EPINET	Euclidean Patterns in Non-Euclidean Tilings
ES	Extended Schläfli symbol for circuits
ICSD	Inorganic Crystal Structure Database
MCN	Molecular coordination number
MOF	Metal-organic framework
MPT	Maximal proper tile
PDB	Protein Databank
RCSR	Reticular Chemistry Structure Resource
SBU	Secondary building unit
TTD	*TOPOS* Topological Database
VDP	Voronoi-Dirichlet polyhedron
VS	Vertex symbol

1. Introduction

At present, the data for more than 400,000 chemical compounds are collected in world-wide crystallographic databases CSD, ICSD, PDB and CrystMet. Processing of such a large amount of information is a great challenge for modern crystal chemistry. Traditional visual analysis of crystal structures becomes insufficient to reveal the common principles of the spatial organization of three-dimensional nets and packings in long series of chemical compounds of various composition and stoichiometry. Rapidly developing interdisciplinary branches of science, such as crystal engineering and supramolecular chemistry, require the development of new computer methods to process and classify the crystallographic information, and to search for general crystallochemical regularities.

When developing the program package *TOPOS* we pursued two main goals:

- to create a computer system that would enable one to perform comprehensive crystallochemical analysis of any crystal structure irrespective of its chemical nature and complexity;
- to implement new methods for crystallochemical analysis of large amounts of chemical compounds to find the regularities in their structure organization in an automated mode.

TOPOS has been developed since 1989 and has several versions being exploited so far. The *MS DOS* versions 3.0, 3.1 and 3.2 were developed until 2003 and now are not supported. The current Windows-based *TOPOS 4.0 Professional* started in 2001 and now is the main program product in the *TOPOS* family. Its periodically updated beta-version is available for free at the *TOPOS* website: http://www.topos.ssu.samara.ru/. It is the version that will be considered in detail and below it will be called *TOPOS* for short.

TOPOS is created using Borland Delphi 7.0 environment and works under *Windows 95/98/Me/NT/2000/XP* operating systems. Its current size is less than 3 M (without topological databases) so it is easily distributed as a self-extracted zipped file. The system requirements are minimal; really *TOPOS* can work on any IBM PC computer under *Windows*. The main file `topos40.exe` is an integrated interactive multiwindow system (Fig. 1) that is based on *DBMS* intended to input, edit, search and retrieve the crystal structure information stored in *TOPOS* external databases. *TOPOS* includes a number of applied programs (Table 1), all of which (except *StatPack*) are integrated into *TOPOS* system.

Table 1: A brief description of *TOPOS* applied programs

Program name	Destination
ADS (Automatic Description of Structure)	Revealing structural groups, determining their composition, orientation, dimensionality and binding in various structure representations
	Calculating topological invariants (coordination sequences, Schläfli and vertex symbols) and performing topological classification
	Constructing molecular VDPs and calculating their geometric characteristics
	Constructing tilings for 3D nets
	Searching for and classifying entanglements of 1D, 2D or 3D extended structures
AutoCN	Identifying and classifying interatomic contacts
	Determining coordination numbers of atoms
	Calculating and storing the adjacency matrix
DiAn	Calculating interatomic distances and bond angles
Dirichlet	Constructing VDPs for atoms and voids
	Calculating geometric characteristics of atom and void domains
	Searching for void positions and channel systems
HSite	Generating positions of hydrogens
IsoCryst	Visualizing crystal structure
	Calculating geometric parameters of crystal structure
IsoTest	Arranging crystal structures into topological and structure types
	Comparative analysis of atomic nets and packings
StatPack	Statistical processing of the data files generated by the programs *Dirichlet* and *ADS*

All *TOPOS* constituents can exchange the data and should usually be applied in a certain sequence when performing a complicated crystallochemical analysis. Scheme 1 shows the logical interconnections within the *TOPOS* system when exchanging the data streams. The main data stream is directed from the top to the bottom of Scheme 1, since all *TOPOS* applied programs use the crystallographic information from *DBMS*. However, *ADS*, *AutoCN*, *Dirichlet*, *HSite* and *IsoTest* programs can produce new data that can be stored in *TOPOS* databases, so there is an inverse data stream.

Crystal structure database in the *TOPOS VER 2.02* format includes five files:

File type	File destination
`*.adm`	contains adjacent matrices of crystal structures *(optional file)*
`*.cmp`	contains chemical formulae of compounds
`*.cd`	contains other data on crystal structures
`*.its`	contains the information on the topology of the graphs of crystal structures *(optional file)*
`*.itl`	contains the information on the topology of atomic sublattices *(optional file)*

DBMS identifies the database using the `*.cmp` file; it is the file that is loaded in the *DBMS* window. Any number of databases can be loaded at once. In addition a number of index files `*.idx` (`'x'` is a letter characterizing the content of the index file) can be created using the *DBMS* ***Distribution*** utility.

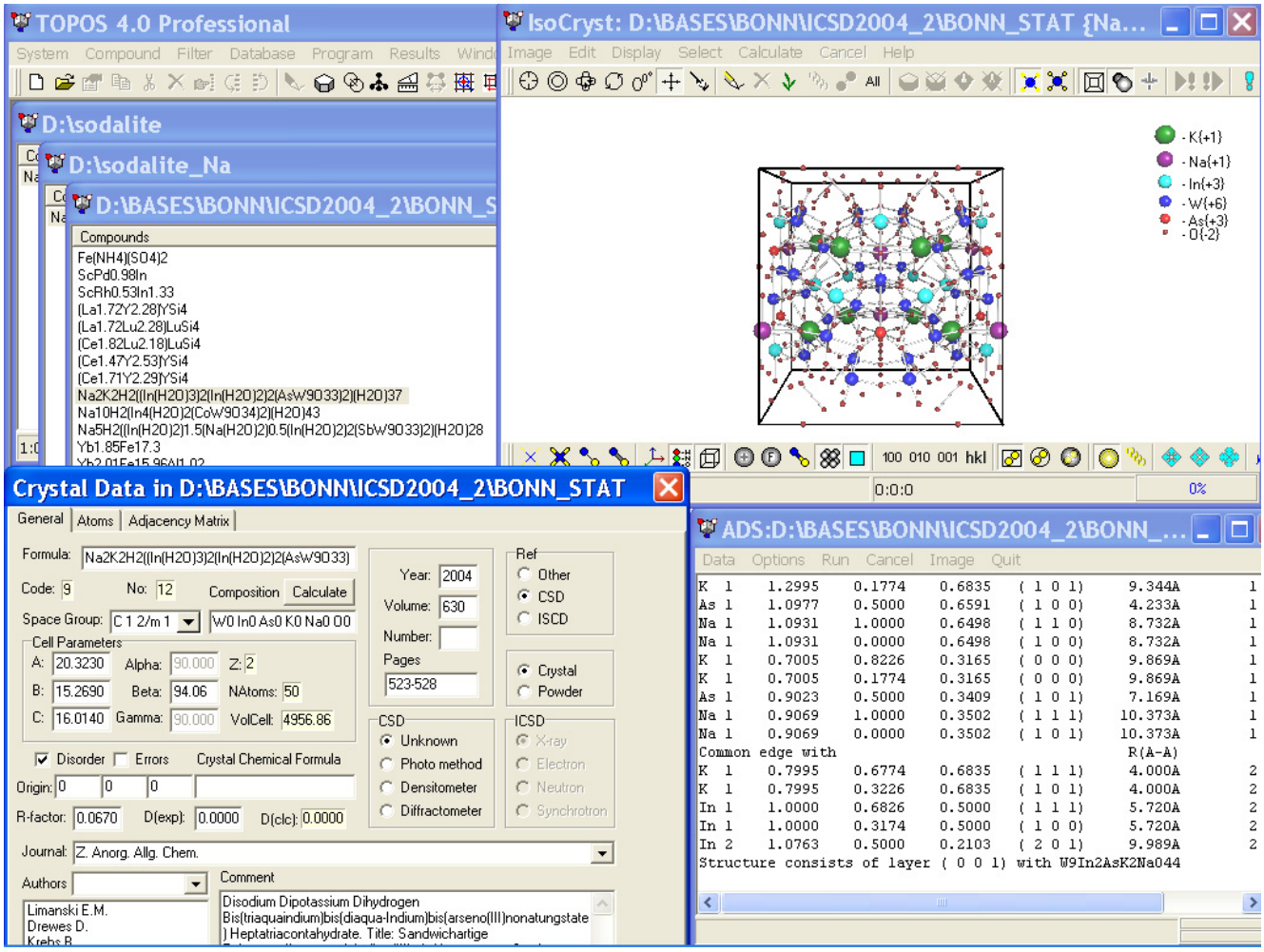
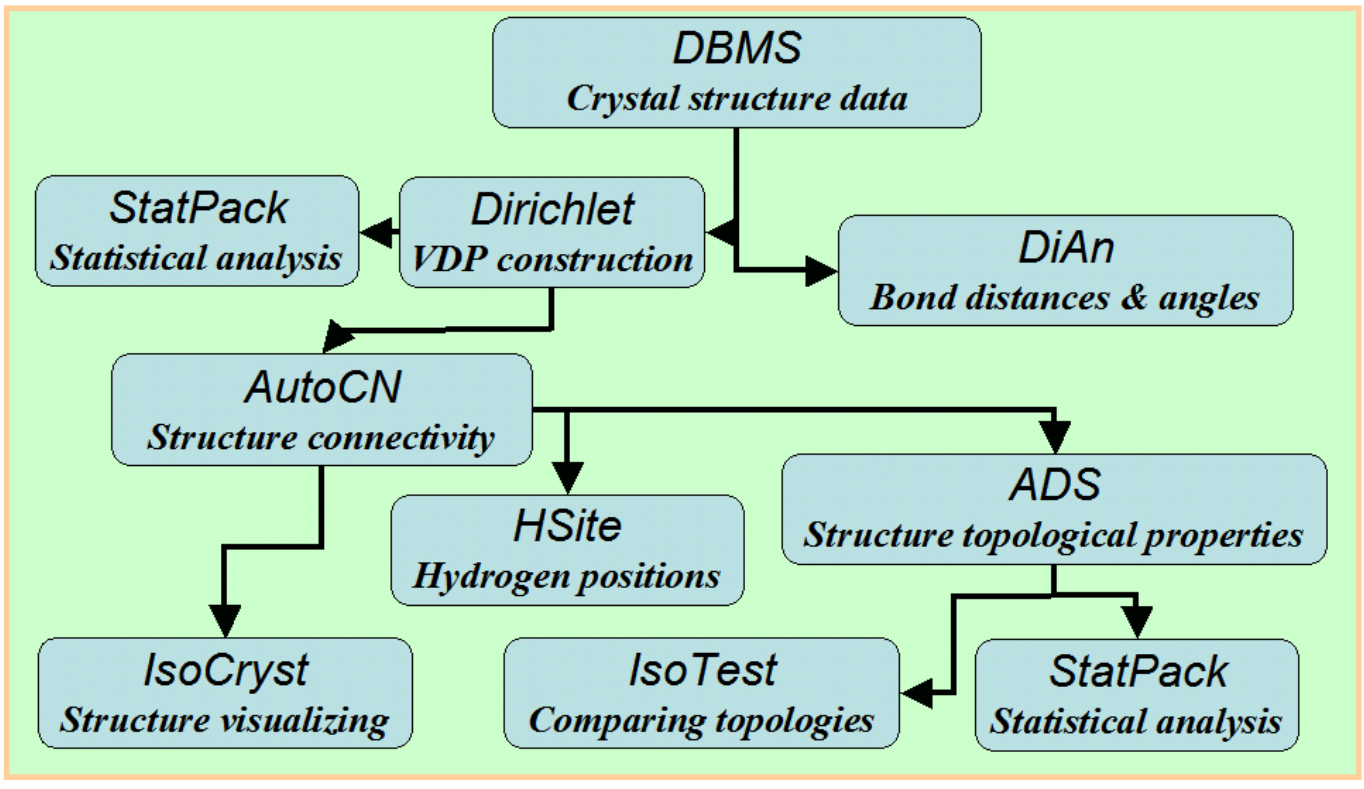


Figure 1: *General view of the program package TOPOS.*



Scheme 1: *Interaction of constituents and main data stream paths within the TOPOS system.*

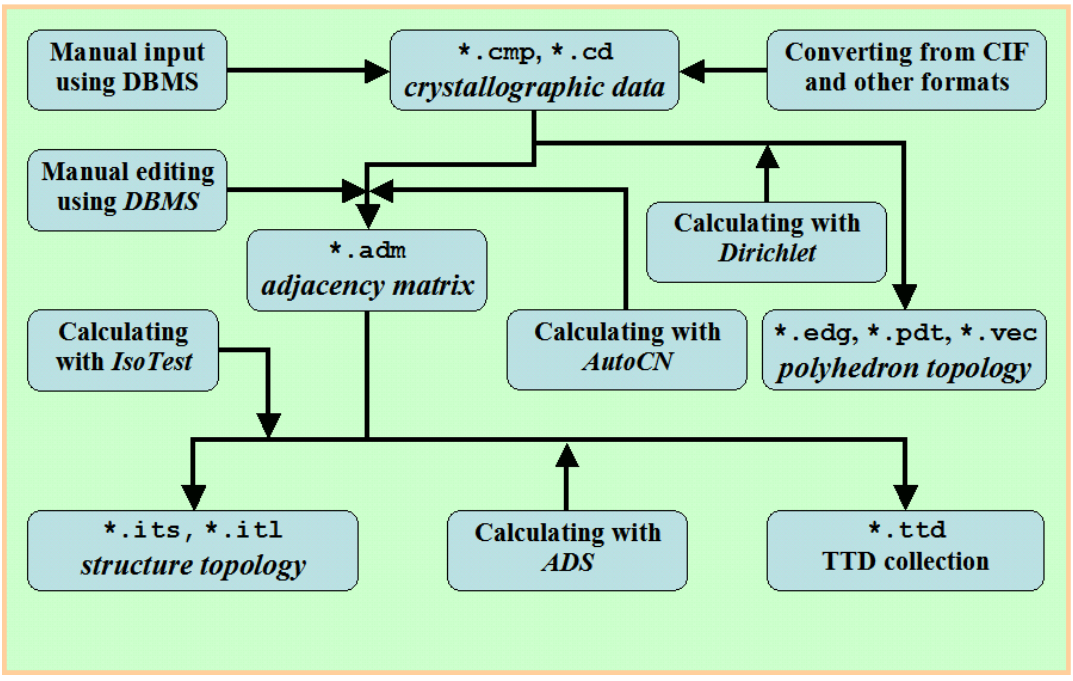
In addition *TOPOS* forms and supports the following auxiliary databases:

- *TTD* collection that is a set of `*.ttd` files in a special binary format containing the information on topological types of simple 2D or 3D nets. The *TTD* collection is used for automatic determining crystal structure topology with the *ADS* program. At present the *TTD* collection includes four databases:

Database	Number of records	Description
`TOPOS&RCSR.ttd`	1606	Data on idealized nets from RCSR,[1] framework zeolites,[2] sphere packings (see, e.g. Sowa & Koch, 2005), and two-dimensional nets
`binary.ttd`	1597	Data on binary framework compounds
`polytypes.ttd`	694	Data on topologies of polytypic close packings, SiC, NiAs, and other layered polymorphs up to 12-layered
`epinet.ttd`	14380	Data on all new topological types of the periodic nets generated within the EPINET project[3]

- library of combinatorial-topological types of finite polyhedra containing the information on the edge nets of polyhedra in the files `*.edg`, `*.pdt`, `*.vec`. This library is used by the *Dirichlet* and *ADS* programs to identify the combinatorial topology of VDPs and tiles.

The methods of inputting the information into the *TOPOS* databases are shown in Scheme 2. The main distinction of the content of the *TOPOS* databases in comparison with other crystallographic databanks is that the 3D graph of interatomic bonds is completely stored in the `*.adm` file. Using this information *TOPOS* can produce other important data on the crystal structure topology. Thus, the main *TOPOS* peculiarity is its orientation to topological characteristics that clarifies its name.



Scheme 2: *Methods to produce data in TOPOS databases.*

2. Topological information in *TOPOS*

[1] http://okeeffe-ws1.la.asu.edu/RCSR/home.htm
[2] http://www.iza-structure.org/databases/
[3] http://epinet.anu.edu.au/

2.1. Adjacency matrix

TOPOS uses the concept of *labelled quotient graph* (Chung *et al*., 1984) to make the infinite 3D periodic graph of crystal structure suitable for computer storage. The adjacency matrix of the labelled quotient graph contained in the `*.adm` file carries all necessary information about the system of interatomic contacts. The format of data for each contact of the basic 'central' atom with a surrounding atom is given below.[4] The **CSym** and **translation** fields contain encoded symmetry operation and translation vector, which transforms the *j*th basic atom into surrounding atom connected with the *i*th central one. This information is sufficient to describe the labelled quotient graph and the topology of the whole net. Other parameters characterize the kind and strength of the contact.

```
record
     i,j:integer numbers of central and surrounding atom
     CSym:integer symmetry code
     translation:array[1..3]of integer translation vector
     m:integer type of the contact
     {m=0 - not a contact
      m=1 - valence bond
      m=2 - specific (secondary) interaction
      m=3 - van der Waals bonding
      m=4 - hydrogen bonding
      m=5 - agostic bonding}
     R,SA:float contact parameters (interatomic distance, VDP solid
     angle, etc.)
end;
```

The program *AutoCN* is intended for automated computing and storing adjacency matrix. Since *TOPOS* can work with periodic nets of various nature, including idealized or artificial nets, *AutoCN* uses several algorithms to determine contacts between nodes of the net.

Three main *AutoCN* algorithms, called **Using Rsds**, **Sectors** and **Distances**, are designed for crystal structures of real chemical compounds and based on constructing ***Voronoi-Dirichlet polyhedra***,[5] VDPs, for all atoms. For applications of VDPs in crystal chemistry see Blatov (2004). The VDP construction uses very effective 'gift wrapping' algorithm (Preparata & Shamos, 1985) of computing a convex hull for a set of image points with coordinates ($2x_i/R_i^2$, $2y_i/R_i^2$, $2z_i/R_i^2$), where (x_i, y_i, z_i) are the Cartesian coordinates of the surrounding atoms, and R_i is the distance from the VDP atom to the *i*th neighbouring atom. In this algorithm for each edge E of face F belonging to the convex hull, the point (P_k) corresponding to the third vertex of a face adjacent to F and joined to it at the same edge is determined from the maximum dihedral angle φ (Fig. 2*a*). Cotangents of φ angles are calculated with the formula

$$\cot\varphi = -\frac{|\boldsymbol{UP}_2|}{|\boldsymbol{UV}|} = -\frac{\boldsymbol{v}_k \cdot \boldsymbol{a}}{\boldsymbol{v}_k \cdot \boldsymbol{n}}, \qquad (1)$$

where $\boldsymbol{n}$ is the unit normal vector to the face F in a half-space containing the VDP atom, and $\boldsymbol{a}$ is the unit vector normal to both E and $\boldsymbol{n}$ (Fig. 2*b*).

As a result, VDP of an atom in the crystal space is a convex polyhedron whose faces are perpendicular to segments connecting the central atom of VDP and the other surrounding atoms (Fig. 3*a*). VDPs of all atoms form Voronoi-Dirichlet partition of crystal space (Fig. 3*b*). Each face divides the corresponding segment by half and ordinarily the face and segment intersect each other. Otherwise (Fig. 3*c*) the surrounding atom is called '***indirect neighbour***' according to O'Keeffe (1979). All the three *AutoCN*

[4] Hereafter a Pascal-like pseudocode is used to describe *TOPOS* algorithms and data structure.
[5] Hereafter all bold italic terms are explained in *TOPOS* Glossary (Appendix).

algorithms consider only the contacts with direct VDP neighbours as potential bonds. The differences are in consequent arranging of the contacts.

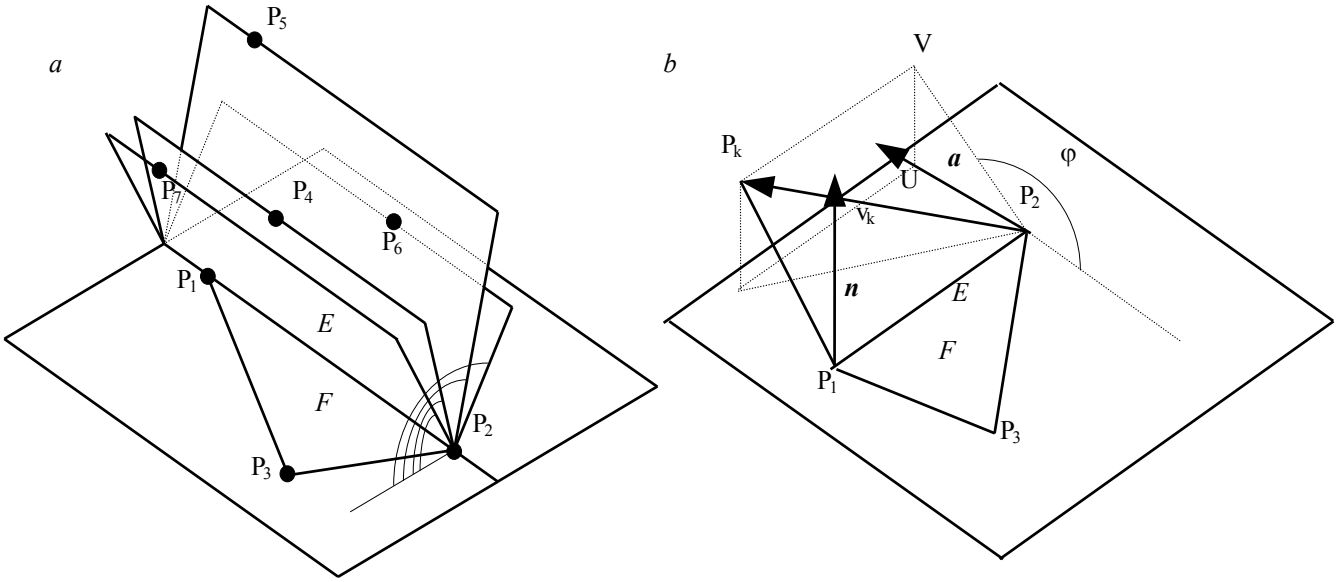


Figure 2: *(a) Determination of a point forming the VDP face (P_6) in the 'gift wrapping' algorithm. The $P_1P_2P_6$ half-plane forms the maximum angle with the $P_1P_2P_3$ (F) half-plane containing previously found points. (b) Calculation of cotφ according to formula (1). P_1P_2 (E) is the VDP edge.*

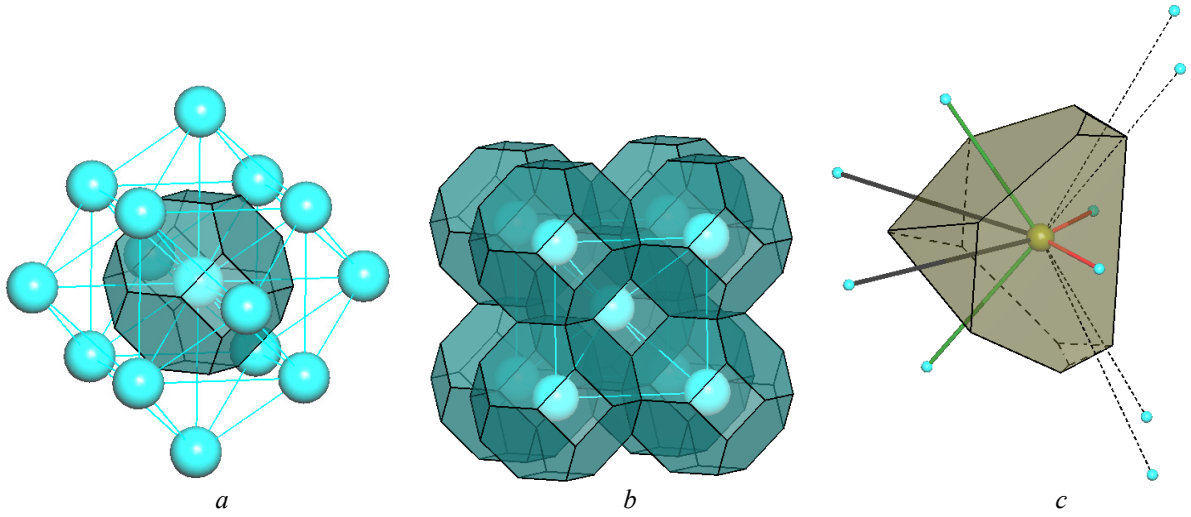


Figure 3: *(a) Voronoi–Dirichlet polyhedron (VDP) and surrounding atoms, (b) Voronoi-Dirichlet partition for a body-centred cubic lattice; (c) VDP and surrounding atoms of an oxygen atom in the crystal structure of ice VIII. Valence, H bond, and non-valence interatomic contacts are coloured red, green, and black, respectively. Indirect contacts are dotted.*

The **Using Rsds** algorithm is rested upon the so-called *method of intersecting spheres* (Serezhkin *et al*., 1997). In this method the interatomic contacts are determined as a result of calculating the number of overlapping pairs of internal and external spheres circumscribed around the centre of either atom of the pair (Fig. 4). Normally, the internal and external spheres have atomic Slater's radius, r_s, and ***radius of spherical domain***, R_{sd}, respectively. If more than one pair of such spheres intersect each other (overlaps Π_2, Π_3 or Π_4) then the contact is assumed to be a chemical bond and is added to atomic CN. If only external spheres overlap, the contact is assumed to be specific, otherwise van der Waals. With additional geometrical criteria the algorithm can separate hydrogen or agostic bonding from specific contacts. In fact, the method of intersecting spheres assumes the shape of the atomic domain to be practically spherical in the crystal structure. This assumption works well for many inorganic compounds, but in the case of organic or coordination compounds it requires considering anisotropy of atomic domains.

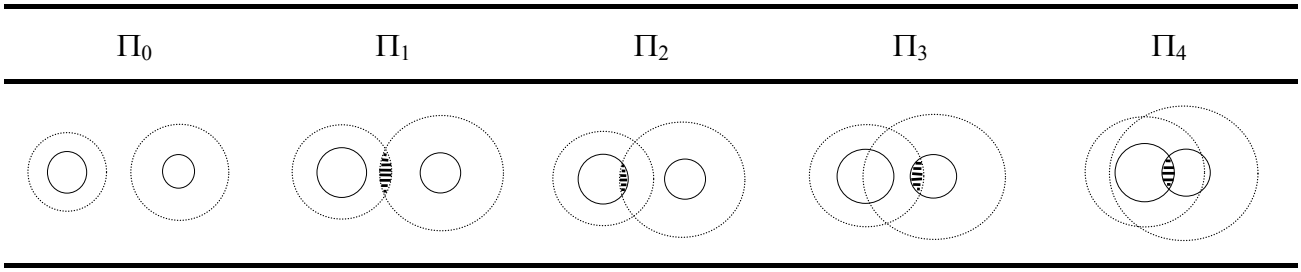


Figure 4: *Schematic representation of basic types of overlaps (Π_n) for atoms within the method of intersecting spheres. The radii of solid and dotted spheres are equal r_s and R_{sd}, respectively. The intersections are shaded of the spheres that causes a given type of overlap. The n value is equal to the number of pair overlaps (Serezhkin et al., 1997).*

The **Sectors** algorithm uses an improved method of intersecting spheres designed by Peresypkina and Blatov (2000) for organic and metal-organic compounds and called *method of spherical sectors*. In this method sphere of R_{sd} radius is replaced with a set of spherical sectors corresponding to interatomic contacts (Fig. 5*a*). The radius (r_{sec}) of the *i*th sector is determined by the formula

$$r_{sec} = \left(\frac{3V_i}{\Omega_i} \right)^{1/3}, \qquad (2)$$

where V_i and Ω_i are volume and solid angle of a pyramid with basal VDP face corresponding to interatomic contacts and with the VDP atom in the vertex (Fig. 5*b*). The **Sectors** algorithm also allows user to reveal non-valence bonding.

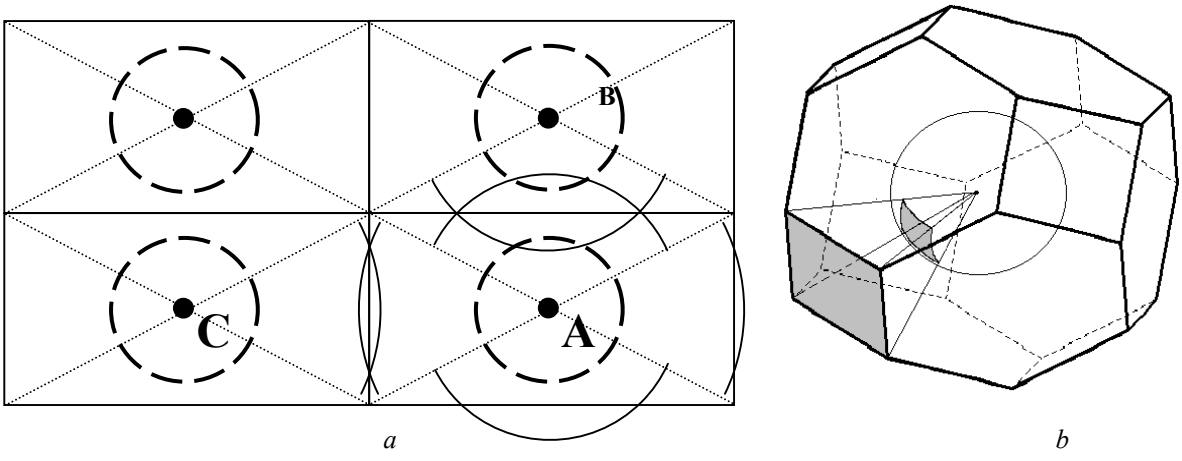


Figure 5: *(a) An example of identification of interatomic contacts with the **Sectors** algorithm in a two-dimensional lattice. Bold lines confine VDPs; dashed lines show boundaries of pyramids (triangles in 2D case) based on the VDP faces corresponding to the direct interatomic contacts. Dashed circles have r_s radius; solid arcs of r_{sec} radius confine spherical sectors and show atomic boundaries in a crystal field. The A and B atoms form a valence contact, to which the triple overlap $r_{sec}(A)$–$r_s(B)$, $r_s(A)$–$r_{sec}(B)$ and $r_{sec}(A)$–$r_{sec}(B)$ corresponds; the contact between A and C atoms is non-valence because the only overlap $r_{sec}(A)$–$r_s(B)$ corresponds to it. (b) VDP of an atom in a body-centred cubic lattice. The solid angle (Ω) of the VDP pyramid based on the shaded face is equal to the shaded segment of the unit sphere cut off by the pyramid with the VDP atom at the vertex and the face in the base.*

The **Distance** algorithm is an attempt to combine the Voronoi-Dirichlet approach and traditional methods that use atomic radii and interatomic distances. The contact between the VDP atom and surrounding atom

is considered valence bonding if the distance between them is shorter that the sum of their Slater's radii increased by a shift to be specified by user (0.3 Å by default).

With these algorithms (**Sectors** by default) user can compute adjacency matrices in an automated mode that is very important for the analysis of large numbers of crystal structures. Their main advantage is independence of the nature of bonding and of kind of interacting atoms; Slater's system of radii is used in all cases. They were tested for all compounds from CSD and ICSD, and showed a good agreement with chemical models.

To work with artificial nets *TOPOS* has two additional algorithms, where no atomic radii and the concept of direct neighbour are used:

Solid Angles, where Ω_i value is the only criterion to select connected net nodes from surrounding ones;

Ranges, where the nodes are considered connected if the distance between them falls into specified range(s); no VDPs are constructed in this case.

The general *AutoCN* procedure with use of one of the VDP algorithms for a crystal structure with `NAtoms` atoms in asymmetric unit is given below. The procedure results in saving `AdjMatr` array containing adjacency matrix.

```
procedure AutoCN(output AdjMatr)
for i:=1 to NAtoms do
  call VDPConstruction(i, output NVDPFaces)
k:=0
for i:=1 to NAtoms do for j:=1 to NVDPFaces[i] do
begin
  k:=k+1
  call CalcContactParam(i, j, output Dist, Omega, Overlap, Direct, HBond, Agostic)
  AdjMatr[k].i:=i
  AdjMatr[k].j:=j
  AdjMatr[k].R:=Dist
  AdjMatr[k].SA:=Omega
  if Omega>OmegaMin then
  begin
    if Method=Solid_Angles then AdjMatr[k].m:=1 else
    if Direct then
    begin
      if (Method=Using_Rsds)or(Method=Sectors) then
        if Overlap=0 then AdjMatr[k].m:=3
        if Overlap=1 then
          if HBond then AdjMatr[k].m:=4 else
            if Agostic then AdjMatr[k].m:=5 else AdjMatr[k].m:=2
        if Overlap>1 then AdjMatr[k].m:=1
      if Method=Distances then
        if Dist<r[i]+r[j]+Shift then AdjMatr[k].m:=1 else AdjMatr[k].m:=0
    end else AdjMatr[k].m:=0
  end else AdjMatr[k].m:=0
end
call StoreInDatabase(AdjMatr)
```

Adjacency matrix is used by all *TOPOS* applied programs; *ADS* and *IsoTest* produce other data for the database derived from the adjacency matrix.

2.2. Reference databases of topological types

The *ADS* program produces textual `*.nnt` (New Net Topology) files that contain important topological invariants of nets and can be converted to binary *TTD* databases. The format of an `*.nnt` file entry is given below. For detailed information on ***coordination sequences***, ***total*** and ***extended Schläfli symbols*** (ES) and ***vertex symbols*** (VS) see Delgado-Friedrichs & O'Keeffe (2005). The CS+ES+VS combination of topological invariants unambiguously determines the topology of any net found in real crystal structures; about additional invariants see part 3.2.1. The binary `*.ttd` equivalents of `*.nnt` files are used as libraries of standard reference nets (topological types) to be compared with the nets in real crystal structures.

An `*.nnt` entry example

```
'$sqc691',
'{6^2;8}{6^4;8^2}{6^5;10}',
'3 8 18 40 65 100 140 184 234 294',
'[6(2).6(2).8(2)]',
'[6(2).6(2).8(2)]',
'4 10 24 44 74 104 144 190 240 296',
'[6.6.6.6.6(2).10(8)]',
'[6.6.6.6.6(2).10(6)]',
'4 12 24 46 72 106 144 190 240 298',
'[6(2).6(2).6(2).6(2).8(2).8(2)]',
'[6(2).6(2).6(2).6(2).8(2).*]',
```

Detailed description:

```
'$sqc691',
```

Name of the record with '$' prefix

```
'{6^2;8}{6^4;8^2}{6^5;10}',
```

Total Schläfli symbol for the whole net: {$6^2$8}{$6^4$$8^2$}{$6^5$10}. In this case the numbers of the three non-equivalent nodes are the same: 1:1:1. Otherwise, indices will be given after each '}' bracket.

```
'3 8 18 40 65 100 140 184 234 294',
```

Coordination sequence (CS)

```
'[6(2).6(2).8(2)]',
```

*Extended Schläfli symbol for **circuits** (ES): [6_2.6_2.8_2]*

```
'[6(2).6(2).8(2)]',
```

*The same for **rings** (VS)*

Similar triples for other non-equivalent nodes

```
'4 10 24 44 74 104 144 190 240 296',
'[6.6.6.6.6(2).10(8)]',
'[6.6.6.6.6(2).10(6)]',
'4 12 24 46 72 106 144 190 240 298',
'[6(2).6(2).6(2).6(2).8(2).8(2)]',
'[6(2).6(2).6(2).6(2).8(2).*]',
```

'' means that there are no rings in this angle, it is equivalent to the '∞' symbol: [6_2.6_2.6_2.6_2.8_2.∞]*

2.3. Topological information on crystal structure representations

The *IsoTest* program forms two kinds of database files. The file `*.its` contains topological invariants (CS+ES+VS) for all possible *net* representations of a given crystal structure. A hierarchical sequence of the crystal structure representations is based on the *complete* representation, where all the contacts stored in the adjacency matrix are taken into account. Each contact (graph edge) has a colour corresponding to its type (the **`m`** field of adjacency matrix), and weight determining by interatomic distance (**`Dist`** field) or solid angle (**`SA`** field). All other representations may be deduced as the subsets of the complete representation by the following three-step algorithm.

(i) Graph edges of the same colour are taken into account, other edges are either ignored or considered irrespective of their weights. In most cases, the chemical interactions of only one type are of interest; as a rule, those are strong bonds. If two or more types of bonds are to be analyzed, the bonds of only one type are to be considered at a given pass of the procedure. Then an array of the weights is formed for all the one-coloured edges.

(ii) The entire array of weights is divided into several groups by a clustering algorithm. *TOPOS* have used a simple approach when two weights belong to the same group if their difference is smaller than a given value. Thus, n distinct coordination spheres are separated in the atomic environment. Then different topologies are generated by successive rejecting the farthest coordination sphere. As a result, n–1 additional representations of the crystal structure are produced from the complete one. It is important that no 'best' representations are chosen at this step, but all levels of interatomic interaction are clearly distinguished for further analysis, depending on the matter in hand.

(iii) Each of the n representations is used to generate a set of subrepresentations according to the scheme proposed by Blatov (2006). Every subrepresentation is unambiguously determined by an arrangement of the set {`NAtoms`} of all atoms from asymmetric unit into four subsets: origin {OA}, removed {RA}, contracted {CA}, and target {TA} atoms. The two operations are defined on the subsets to derive a graph of the subrepresentation from the graph of an initial ith representation: contracting an atom to other atoms keeping the local connectivity, when the atom is suppressed, but all graph paths passing through it are retained (Figs. 6*a*,*b*), and removing an atom together with all its bonds (Figs. 6*c*,*d*). The four-subset arrangement is determined by the role of atoms in those operations. Namely, origin atoms form a new net that characterizes the subrepresentation topology; removed atoms are eliminated from the initial net by the removing operation; contracted atoms merge with target atoms, passing the bonds to them.

All the sets {OA}, {RA}, {CA}, and {TA} form a *collection* ({OA}, {RA}, {CA}, {TA}) that, together with the initial representation, unambiguously determines the subrepresentation topology (Figs. 6*a-d*). With the concept of collection, the successful enumeration of the significant subrepresentations becomes easily formalizable as a computer algorithm implemented into *IsoTest*. Firstly, any collection has a number of properties reflecting the crystal structure relations that can be formulated in terms of set theory.

(i) $\{OA\}\cap\{RA\}=\varnothing$; $\{OA\}\cap\{CA\}=\varnothing$, $\{RA\}\cap\{CA\}=\varnothing$, because an atom cannot play more than one role in the crystal structure.
(ii) $\{OA\}\cup\{RA\}\cup\{CA\}=\{$`NAtoms`$\}$, *i.e.* every atom must have a crystallochemical role.
(iii) $\{OA\}\neq\varnothing$, other sets may be empty. This property arises because only the origin atoms are nodes in the graph of the crystal structure subrepresentation; other atoms determine the graph topology. Obviously, the collection ({OA}, ∅, ∅, ∅) means that {OA}={`NAtoms`}; it describes the initial representation.
(iv) $\{TA\}\subseteq\{OA\}$, because the target atoms are always selected from the origin atoms; unlike other origin atoms they are the centres of complex structural groups.
(v) $\{TA\}\neq\varnothing \Leftrightarrow \{CA\}\neq\varnothing$, because the target and contracted atoms together form the structural groups.

Secondly, the collections, together with the topological operations, map onto *all* the crystal structure transformations applied in crystallochemical analysis. Namely, origin atoms correspond to the centres of structural groups in a given structure consideration. If a structural group has no distinct central atom, a pseudo-atom (PA) coinciding with group's centroid should be added to the {NAtoms} set; this case is typical to the analysis of molecular packings. Removed atoms are atoms to be ignored in the current crystal structure representation, as atoms of interstitial ions and molecules in porous substances or, say, alkali metals in framework coordination compounds. Contracted atoms, together with target atoms, form complex structural groups, but the contracted atoms are not directly considered; they merely provide the structure connectivity whereas the target atoms coincide with the groups' centroids. The difference between origin and target atoms is that the target atoms always correspond to *polyatomic* structural groups whereas the origin atoms symbolize all structural units, both mono- and polyatomic.

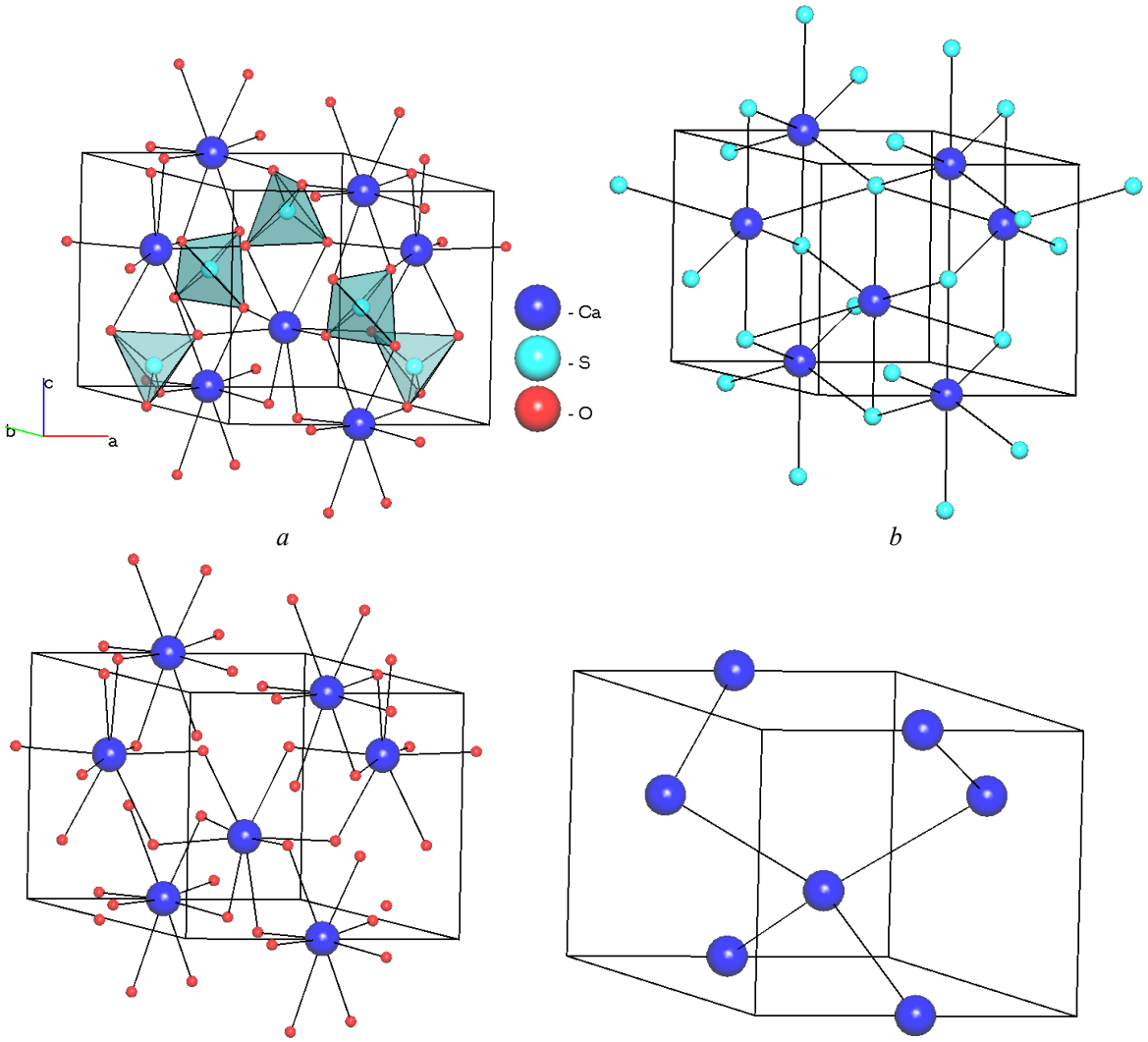


Figure 6: *γ-$CaSO_4$ crystal structure: (a) complete representation ({Ca, S, O}, ∅, ∅, ∅), and its subrepresentations (b) ({Ca, S}, ∅, {O}, {S}) with origin Ca and S atoms, contracted oxygen atoms, and target sulfur atoms (the **sma**[6] topology); (c) ({Ca, O}, {S}, ∅, ∅) with origin Ca and O atoms, and removed sulfur atoms; (d) ({Ca}, {S}, {O}, {Ca}) with origin and target Ca atoms, removed sulfur atoms, and contracted oxygen atoms (the **qtz** topology).*

[6] The bold three-letter codes indicate the net topology according to the RCSR nomenclature (http://okeeffe-ws1.la.asu.edu/RCSR/home.htm).

If, say, there are two atoms of different colours, A and B, {A, B}={NAtoms}, the following four subrepresentations are possible for the initial representation ({A, B}, ∅, ∅, ∅):

(i) ({A}, {B}, ∅, ∅), *i.e.* the subnet of A atoms;
(ii) ({A}, ∅, {B}, {A}), *i.e.* the net of A atoms with the A–B–A bridges (B atoms are spacers);
(iii) ({B}, {A}, ∅, ∅); (iv) ({B}, ∅, {A}, {B}) are the same nets of B atoms.

IsoTest enumerates all possible collections and successively writes down them into *.its file in the following format:

```
OA, RA, CA, TA: array of integer
  atomic numbers for atoms in the {OA}, {RA}, {CA}, {TA} sets
CS, ES, VS: array of integer topological invariants for all OA atoms
...
```

Another *IsoTest* algorithm enables user to compute topological invariants for sublattices of, generally speaking, non-bonded atoms, and to save them in the *.itl file. Actually, the *.itl file contains the topological information on all possible packings of atoms. There are two principal distinctions in this algorithm in comparison with the analysis of nets:

(i) adjacency matrix is calculated using the **Solid Angles** algorithm because no real chemical bonds, but packing contacts, are analyzed;

(ii) all atoms in the collection are considered origin or removed, no contraction is used because of the same reason.

Thus, in the case of an AB compound three *packing* representations ({OA}, {RA}) will be considered: ({A}, {B}), ({B}, {A}) and ({A, B}, ∅). The formats of *.its and *.itl files are similar, but there are no **CA** and **TA** arrays in the *.itl file.

2.4. Library of combinatorial types of polyhedra

Two *TOPOS* programs, *Dirichlet* and *ADS*, can store the data on polyhedral units in a library consisting of three files: *.pdt (polyhedron name and geometrical parameters); *.edg (data on polyhedron edges in the format: **V1, V2: integer**, where V1 and V2 are the numbers of polyhedron vertices); *.vec (Cartesian coordinates of vertices and face centroids). Using the polyhedron adjacency matrix from the *.edg file *Dirichlet* and *ADS* can unambiguously identify combinatorial topology of VDPs and tiles. A standard algorithm of searching for isomorphism of two finite ordinary graphs is used for this purpose.

3. Basic algorithms of crystallochemical analysis in *TOPOS*

In accordance to the content of databases there are two principal ways of crystallochemical analysis in *TOPOS*. They can be conditionally called *geometrical* and *topological*, because the former one rests upon the ordinal crystallographic data from *.cd file (cell dimensions, space group, atomic coordinates), whereas the latter one uses the topological information from *.adm, *.its, *.itl *.ttd, *.edg files. As is seen from the previous part these two ways are not completely independent, because all the topological data are initially produced from crystallographic information. However, these two methods depend on different algorithms, and we need to describe them separately.

3.1. Geometrical analysis: general scheme

Here we consider in detail only original *TOPOS* features that distinguish it from well known crystallochemical software such as *Diamond*, *Platon*, ICSD or CSD tools. In addition, the *IsoCryst* and *DiAn* programs let user compute all standard geometrical parameters (interatomic distances, bond and torsion angles, RMS lines and planes, *etc*.) with ordinal algorithms. The general scheme of geometrical analysis of a crystal structure is shown in Scheme 3.

3.1.1. Computing atomic and molecular Voronoi-Dirichlet polyhedra

Geometrical analysis in *TOPOS* is based on VDP as an image of an ***atomic domain*** in the crystal field and on Voronoi-Dirichlet partition as an image of crystal space that is a good approach even in the case of complex compounds (Blatov, 2004). The main advantage of this approach over the traditional model of a spherical atom is its independence of any system of atomic radii and validity for describing chemical compounds of different nature, from elementary substances to proteins. The programs *Dirichlet* and *IsoCryst* compute the following geometrical and topological VDP parameters, each of which has a clear physical meaning (Blatov, 2004; Table 2):

- VDP volume (V_{VDP}) and R_{sd}.
- VDP dimensionless normalized second moment of inertia (G_3), generally defined as:

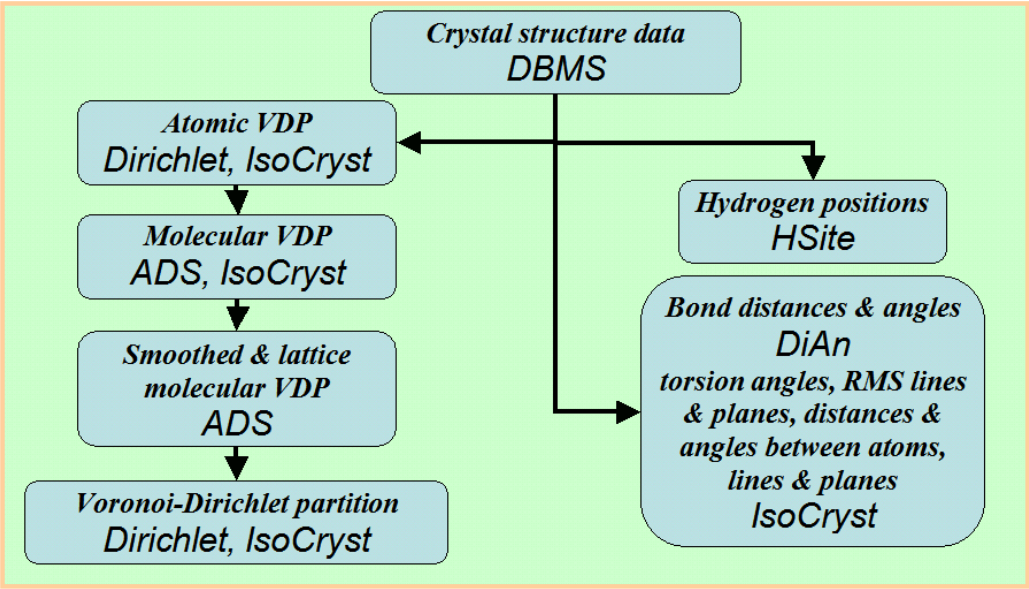
$$G_3 = \frac{1}{3}\frac{\int_{VDP} R^2 dV_{VDP}}{V_{VDP}^{\frac{5}{3}}}, \qquad (3)$$

however, *Dirichlet* uses a simpler formula for an arbitrary (not necessarily convex) solid that can be subjected to simplicial subdivision:

$$G_3 = \frac{1}{3}\frac{\sum_j V_j I_j}{V_{VDP}^{\frac{5}{3}}}, \qquad (4)$$

where summation is performed over all simplexes, V_j is the volume of the jth simplex, and I_j is the normalized second moment of inertia of a simplex with respect to the centre of the VDP:

$$I = \frac{4}{5}\| \bar{v} \|^2 + \frac{1}{20}\sum_{k=0}^{3}\| v_k \|^2 . \qquad (5)$$



Scheme 3: *Geometrical analysis of a crystal structure in TOPOS.*

In (5), summation is performed over all simplex vertices, $\|v_k\|$ is the norm of the radius vector of the kth vertex of the simplex, and P$\overline{v}$P is the norm of the radius vector of the simplex centroid in the coordinate system with the origin in the centre of the VDP.

- Solid angles of VDP faces (Ω_i) to be computed according to Fig. 5.
- Displacement of an atom from the centroid of its VDP (D_A).
- Number of VDP faces (N_f).

A number of parameters of Voronoi-Dirichlet partition to be computed with *Dirichlet* are crucial at crystallochemical analysis (Table 2):

- Standard deviation for 3D lattice quantizer (Convay and Sloane, 1988):

$$\langle G_3 \rangle = \frac{1}{3} \frac{\frac{1}{NAtoms} \sum_{i=1}^{NAtoms} \int_{VDP(i)} R^2 dV_{VDP(i)}}{\left\{ \frac{1}{NAtoms} \sum_{i=1}^{NAtoms} V_{VDP(i)} \right\}^{\frac{5}{3}}}, \quad (6)$$

or, with (4)

$$\langle G_3 \rangle = \frac{1}{3} \frac{\frac{1}{NAtoms} \sum_{i=1}^{NAtoms} \sum_{j} V_j I_j}{\left\{ \frac{1}{NAtoms} \sum_{i=1}^{NAtoms} V_{VDP(i)} \right\}^{\frac{5}{3}}}, \quad (7)$$

i.e. $\langle G_3 \rangle$ is averaged over G_3 values of all inequivalent atomic VDPs.

- Coordinates of all VDP vertices and lengths of VDP edges.
- Other geometrical parameters of VDP vertices and edges important at the analysis of voids and channels (see part 3.2.2).

Table 2: *Physical meaning of atomic VDP, molecular VDP and Voronoi-Dirichlet partition parameters*

Parameter	Dimensionality	Meaning
Atomic VDP parameters		
V_{VDP}	Å^3	Relative size of atom in the crystal field
R_{sd}	Å	Generalized crystallochemical atomic radius
G_3	Dimensionless	Sphericity degree for nearest environment of the atom; the less G_3, the closer the shape of coordination polyhedron to sphere
Ω_i	Percentage of 4π steradian	Strength of atomic interaction
D_A	Å	Distance between the centres of positive and negative charges in the atomic domain
N_f	Dimensionless	Number of atoms in the nearest environment of the VDP atom
Molecular VDP parameters		
V_{VDP}(mol)	Å^3	Relative size of secondary building unit in the crystal field
R_{sd}(mol)	Å	Effective radius of secondary building unit
G_3(mol)	Dimensionless	Sphericity degree of secondary building unit
MCN, Number of faces of smoothed molecular VDP	Dimensionless	Number of SBUs contacting with a given one
Ω_i^{mol}	Percentage of sum of Ω_i^{mol}	Strength of intermolecular interaction
Number of faces of lattice molecular VDP	Dimensionless	Number of SBUs surrounding a given one in idealized packing of spherical molecules
Voronoi-Dirichlet partition parameters		
$\langle G_3 \rangle$	Dimensionless	Uniformity of crystal structure
Coordinates of VDP vertices	Fractions of unit cell dimensions	Coordinates of void centres
Lengths of VDP edges	Å	Lengths of channels between the voids

Uniting atomic VDPs *TOPOS* constructs secondary building units in the form of ***molecular VDP*** (Figs. 7*a*-*c*). Molecular VDP is always non-convex, however, VDPs of all secondary building units (SBU) in the crystal structure still form the Voronoi-Dirichlet partition of the crystal space. The program *IsoCryst* visualizes molecular VDPs, and the program *ADS* determines the following parameters (Table 2):

- Molecular VDP volume (as a sum of volumes of atomic VDPs), V_{VDP}(mol) and R_{sd}(mol).
- Normalized second moment of inertia of molecular VDP, G_3(mol), to be computed according to (4), but the summation is provided over simplexes of all atomic VDPs composing the molecular VDP, and the centroid of the molecule is taken as origin.
- Molecular coordination number (MCN) as a number of ***molecular VDP faces***.
- Solid angles of molecular VDP faces (Ω_i^{mol}) to be computed by the formula (8)

$$\Omega_i^{mol} = \frac{\sum_j \Omega_{ij}}{\Omega_\Sigma} \cdot 100\%, \qquad (8)$$

where Ω_{ij} are solid angles of the ***molecular VDP facets*** composing the *i*th molecular VDP face; $\Omega_\Sigma = \sum_i \sum_j \Omega_{ij}$ is the sum of solid angles of all nonbonded contacts formed by atoms of the molecule.

- Cumulative solid angles corresponding to different kinds of intermolecular contacts in MOFs:

Valence solid angles of a ligand (Ω_L^V) and a complex ($\Omega_{L\Sigma}^V$) to be calculated as

$$\Omega_L^V = \sum_i \Omega\left(M - X_i\right), \qquad (9)$$

where valence contacts between the complexing M atom and donor X_i atoms of a ligand L were only taken into consideration, and

$$\Omega_{L\Sigma}^V = \sum_I \left(\Omega_L^V\right)_I, \qquad (10)$$

where all ligands connected with the M atom are included in the sum.

Total solid angles of a ligand (Ω_L^T) and a complex ($\Omega_{L\Sigma}^T$):

$$\Omega_L^T = \sum_i \Omega\left(M - X_i\right), \qquad (11)$$

$$\Omega_{L\Sigma}^T = \sum_I \left(\Omega_L^T\right)_I, \qquad (12)$$

where, unlike (9), the index *i* enumerates *all* (including non-valence) contacts *VDP atom–ligand*, even if the ligand is *non-valence* bonded with the complexing atom and only shields it, while the index *I*, as in (10), enumerates all the ligands in complex, which are *valence* bonded with the complexing atom.

Agostic solid angles of a ligand (Ω_L^{ag}) and a complex ($\Omega_{L\Sigma}^{ag}$). These values are to be calculated by the formulae analogous to (11) and (12), but with merely the solid angles of atomic VDPs corresponding to agostic contacts M…H–X.

Residual solid angles of a ligand ($\delta = \Omega_L^T - \Omega_L^V$) and a complex ($\Delta = \Omega_{L\Sigma}^T - \Omega_{L\Sigma}^V$).

In addition to molecular VDPs the *ADS* program constructs two types of VDPs for SBU centroids:

(i) The ***Smoothed molecular VDP*** is constructed by flattening the boundary surfaces of a molecular VDP (Fig. 7*d*). Smoothed molecular VDPs characterize the local topology of molecular packing and occasionally do not form a partition of space.

(ii) The ***Lattice molecular VDP*** is constructed by using molecular centroids only (Fig. 7*e*). Lattice molecular VDPs characterize the global topology of a packing as a whole and form a partition of space, but the number of faces of such a VDP is not always equal to MCN.

In both cases the only VDP parameter, number of faces, has clear crystallochemical meaning (Table 2).

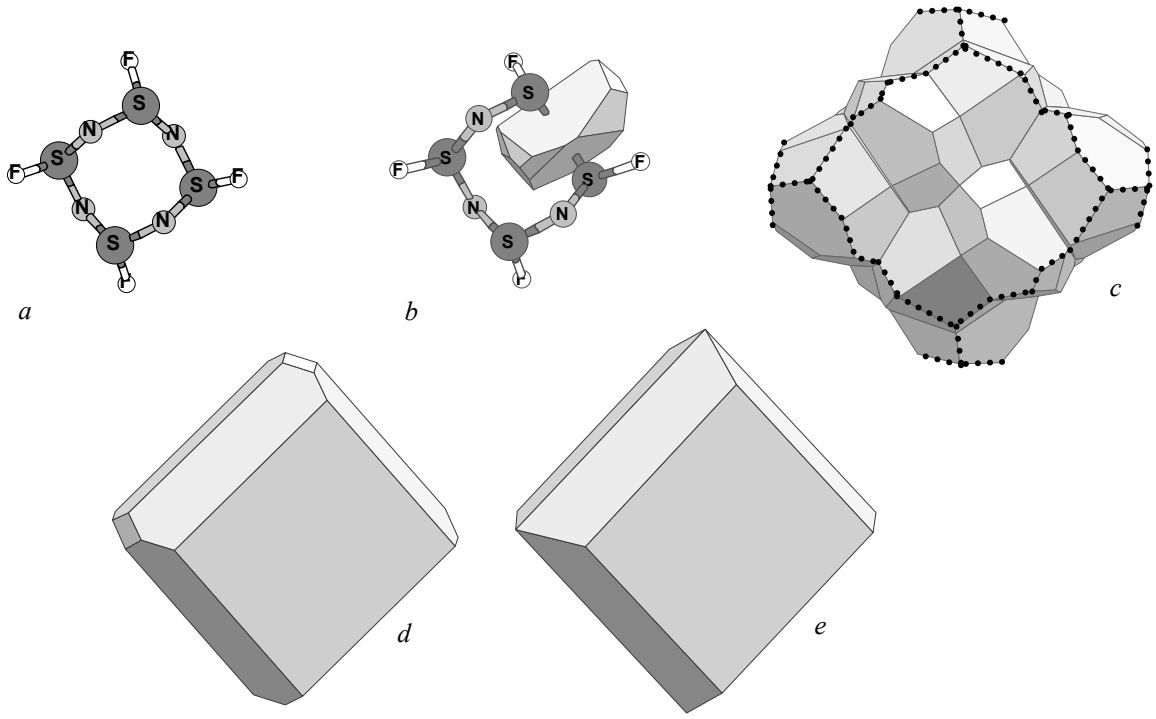


Figure 7: *(a) A molecule $N_4S_4F_4$; (b) VDP of a nitrogen atom; (c) molecular VDP (dotted lines confine boundary surfaces); (d) smoothed and (e) lattice molecular VDPs.*

3.1.2. Generating hydrogen positions

Parameters of atomic VDPs are used in the program *HSite* intended for the calculation of the coordinates of H atoms connected to X atoms (X = B, C, N, O, Si, P, S, Ge, As, Se) depending on their nature, hybridization type and arrangement of other atoms directly non-bonded with the X atoms. In comparison with known similar programs *HSite* has some additional features:

(i) At the determination of the hybridization type of an atom X the Me…X contacts of different type (σ or π) between metal (Me) and X atoms are taken into account.

(ii) During the generation of H atoms in groups with rotational degrees of freedom, the search for an optimal orientation of the group is fulfilled depending on the arrangement and size of the surrounding atoms. In turn, the sizes of these atoms are approximated by their R_{sd} values. In the determination of the optimal orientation the effects of repulsion in H…H contacts are considered and the possibility of the appearance of hydrogen bonds O(N)–H …O(N) is taken into account.

The *HSite* algorithm includes the following steps:

(i) Searching for X atoms, which can be potentially linked with hydrogen atoms.

(ii) Determination of the hybridization (*sp*, sp^2 or sp^3) of these atoms in accordance with the following criteria:

- B, Si and Ge atoms may have the sp^3 hybridization only.
- O, P, S, As and Se atoms may have the sp^2 or sp^3 hybridization only.
- C and N atoms may have any type of hybridization.

- Bonds with metal atoms are taken into account at the determination of hybridization only if they form σ-bonds with X atoms. *HSite* automatically determines the type of Me–X bonds (σ or π) using the following criterion: a pair of X atoms is involved into a π bonding with a Me atom if they are also linked together, *i.e.* there is a triple 

.
- The types of hybridization are distinguished depending on the parameters of valence bonds formed by X atoms with other L atoms:

Total number of X–L bonds	Number of bonds with L=C, N, O, S, Se	Numerical criterion	Hybridization
any	0	none	sp^3
1	1	$R(X\text{-}L) \le R_{max}(sp)$	sp
1	1	$R(X\text{-}L) \le R_{max}(sp^2)$	sp^2
1	1	$R(X\text{-}L > R_{max}(sp^2)$	sp^3
2	1, 2	$\angle L\text{–}X\text{–}L \ge \angle_{min}(sp)$	sp
2	1, 2	$R(X\text{-}L1)+R(X\text{-}L2) < R_{\Sigma}(sp^3)$	sp^2
2	1, 2	$R(X\text{-}L1)+R(X\text{-}L2) \ge R_{\Sigma}(sp^3)$	sp^3
3	1, 2, 3	$\angle L1\text{–}X\text{–}L2 + \angle L1\text{–}X\text{–}L3 + \angle L2\text{–}X\text{–}L3 > \angle_{\Sigma}(sp^2)$	sp^2
3	1, 2, 3	$\angle L1\text{–}X\text{–}L2 + \angle L1\text{–}X\text{–}L3 + \angle L2\text{–}X\text{–}L3 \le \angle_{\Sigma}(sp^2)$	sp^3

The criteria $R_{max}(sp)$, $R_{max}(sp^2)$, $R_{\Sigma}(sp^3)$ have the default values 1.30, 1.40 and 2.90 Å for X, L= C, N and O, respectively. If X or L atom is of the 3rd or the 4th period, then the $R_{max}(sp^2)$ criterion is increased by 0.4 or 0.5 Å, respectively, and $R_{\Sigma}(sp^3)$ is increased by 0.8 or 1.0 Å. If a boron atom participates in the bond, all values increase by 0.11 Å.

(iii) The site symmetry of X and L positions is taken into account. If necessary, the type of hybridization of X atom and the number of hydrogen atoms to be added are corrected. For example, if the above mentioned criteria show that a carbon atom is in sp^3 hybridization and should form a methyl group, but its site symmetry is C_2, then its true hybridization is assumed to be sp^2 and it really forms a planar CH_2 group.

(iv) Positions of hydrogen atoms are determined with the following geometric criteria:
- Bond angles ∠H-X-H depend only on the type of hybridization of the X atom and are equal to 180, 120 and 109.47° for *sp*-, sp^2 and sp^3 hybridization, respectively.
- For the sp^2 hybridization in the group L_2XH the condition ∠L1–X–H = ∠L2–X–H must hold.
- For the sp^3 hybridization in the group L_2XH_2 two additional hydrogen atoms must lie in the plane perpendicular to the plane passing through the L1, L2 and X atoms. In the case of the group L_3XH the condition ∠L1-X-H = ∠L2-X-H = ∠L3-X-H must hold.
- Lengths of the bonds O-H, N-H and C-H are equal 0.96, 1.01 and 1.09 Å by default and may be changed. If the X atom is of the 3rd or the 4th period the bond length will be additionally increased by 0.4 or 0.5 Å, respectively. For example, the length of Se–H bond will be 1.46 Å.
- If the atomic group has rotational degrees of freedom, its optimal orientation is searched in the following way: the group rotates with a small step (5° by default), for each orientation the minimum distance (R_{min}) is found from hydrogen atoms of the group to other atoms except of the atom X itself, normalized by the R_{sd} values for these atoms. The orientation with maximum R_{min} assumes to be optimal. For isolated groups (H_2O, NH_4^+, CH_3^-, *etc*.) all possible orientations of the primary axis of inertia are additionally verified by scanning an independent region of the spherical coordinate system; the spherical coordinates φ and θ vary also with the 5° step. If the H bonds are considered, they take priority at the determination of the orientation. The conditions $R(H\ldots X) \le R_{max}(HBond)$ and ∠X–H…X >

$\angle_{min}$(HBond) are used for distinguishing H bonds. A mandatory condition during searching for the orientation is that the distances between hydrogen atoms and other atoms, except the atoms participating in H bonds, must be more than 2 Å (by default). If this condition cannot be obeyed, the program error '**Atom X is invalid**' is generated. The orientation of bridge groups XH_n binding several metal atoms is a special case. At that the orientation of the primary axis of inertia of the group is considered passing through the centroid of the set of metal atoms and through the X atom itself. The exception is the planar CH_3^+ cation whose orientation may be different taking into account the aforesaid criteria.

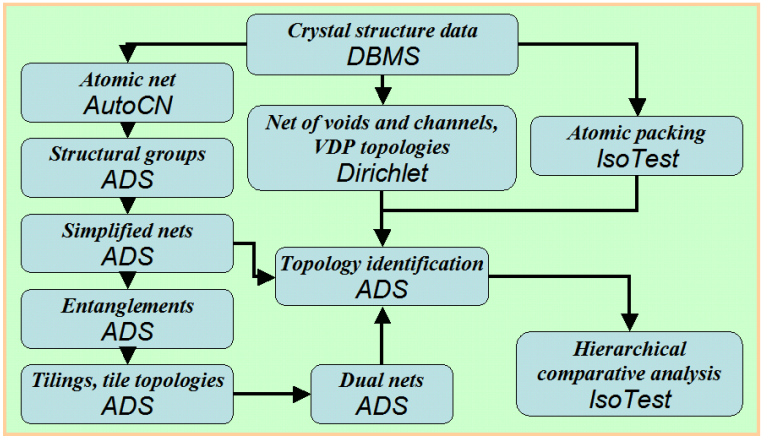
- Boron atoms are assumed to be in the composition of carboran or borohydride ions. The generation of hydrogens is not provided for boranes.

(v) If there are 'pseudo-bonds' Me–X the parameter R_{max}(Me) (5 Å by default) may be useful which corresponds to maximum allowable length of the Me–X bonds to be considered at the determination of the geometry and orientation of the XH_n group. To avoid the 'pseudo-bonds' the R_{max}(Me) may be decreased.

(vi) By default all groups assume to be electroneutral; the valence of the X atoms supposes to be standard and equal to 8 minus number of corresponding group of Periodic Table. If a group is an ion (for example, $X\text{-}NH_3^+$ or OH^-), it may be taken into account by setting corresponding *HSite* options ('**Hydroxide/amide-anions**' or '**Hydroxonium/ammonium-cations**').

3.2. Topological analysis: general scheme

Topological analysis is the main *TOPOS* destination; many modern methods have recently been implemented, and new features appear every year. Below the general scheme of the analysis (Scheme 4) and basic algorithms are considered.



Scheme 4: *Topological analysis of a crystal structure in TOPOS.*

As follows from Scheme 4 there are three representations of crystal structure in *TOPOS*: as an atomic net, as a net of voids and channels, and as an atomic packing. The main branch of the scheme begins with generating atomic net as a labelled quotient graph (part 2.1). The subsequent analysis should be performed with program *ADS*.

3.2.1. Analysis of atomic and molecular nets

To analyze the adjacency matrix of the labelled quotient graph *ADS* uses the sets of origin {OA}, removed {RA}, contracted {CA}, and target {TA} atoms (part 2.3) to be specified by user. There are two modes of the analysis: **Atomic net** ({OA}≠∅) and **Molecular net** ({OA}=∅). The algorithm of the first mode consists of the following steps:

(i) All {RA} are removed from the adjacency matrix.

```
procedure Remove_RA(output AdjMatr)
for i:=1 to NAtoms do
begin
 if Atoms[i] ∈ {RA} then atom must be removed
  repeat
   looking for AdjMatr[k1].i=i or AdjMatr[k1].j=i
   AdjMatr[k1].m:=0 'not a contact' flag
  until no AdjMatr[k1].i=AdjMatr[k].j or AdjMatr[k1].j=i
end
```

(ii) All {CA} form ligands.

```
procedure Form_Ligands(output Ligands)
for i:=1 to NAtoms do
begin
 if Atoms[i]∈{CA} and Atoms[i]∉{Ligands} then atom forms new ligand
 begin
  new Ligands[j]
  add Atoms[i] to Ligands[j]
  for Atoms[k] ∈ Ligands[j] do
  repeat
   looking for AdjMatr[k1].i=k
   if Atoms[AdjMatr[k1].j]∈{CA} then add Atoms[AdjMatr[k1].j] to Ligands[j]
  until no AdjMatr[k1].i=k
 end
end
```

(iii) All {CA} are contracted to {TA}. A simplified net is obtained as a result.

```
procedure Contract_CA_to_TA(output AdjMatr)
for i:=1 to NAtoms do
begin
 if Atoms[i] ∈ {TA} then target atom is found
  repeat
   looking for AdjMatr[k].i=i, i.e. the record corresponding to Atoms[i]
   if Atoms[AdjMatr[k].j] ∈ {CA} then surrounding atom must be contracted
    repeat
     looking for AdjMatr[k1].i=AdjMatr[k].j
     AdjMatr[k1].i:=AdjMatr[k].i
     looking for AdjMatr[k2].j=AdjMatr[k].j
     AdjMatr[k2].j:=AdjMatr[k].i
    until no AdjMatr[k1].i=AdjMatr[k].j
   delete AdjMatr[k]
  until no AdjMatr[k].i=i
end
```

The second mode differs from the first one by additional procedure of determining molecular units to be fulfilled after the first step. In this case initially {OA}={CA}={TA}=∅, but there should be at least two different kinds of bond in adjacency matrix: *intramolecular* and *intermolecular*. A typical situation is when the intramolecular bonds are valence (**`AdjMatr[k].i=1`**) and intermolecular bonds are hydrogen, specific or/and van der Waals (**`AdjMatr[k].i=2,3,4`**). As a result of the additional (i*a*) step, all atoms fall into {CA} set, and molecular centroids ('pseudo-atoms', PA) are input into {OA} and {TA} sets. Subsequent passing the steps (ii) and (iii) results in the connected net of molecular centroids.

(i*a*) Searching for molecular units (**Molecular net** mode).

```
procedure Form_Molecules(output Molecules, AdjMatr)
for i:=1 to NAtoms do
```

```
begin
 if Atoms[i]∉{CA} and Atoms[i]∉{Molecules} then atom forms new molecule
 begin
  new Molecules[j]
  add Atoms[i] to Molecules[j]
  add Atoms[i] to {CA}
  for Atoms[k] ∈ Molecules[j] do
  repeat
   looking for AdjMatr[k1].i=k
   if AdjMatr[k1].m=1 then
   begin
    add Atoms[AdjMatr[k1].j] to Molecules[j]
    add Atoms[AdjMatr[k1].j] to {CA}
   end
  until no AdjMatr[k1].m=1
  call Calc_Centroid(Molecules[j], output PA[j])
  add PA[j] to {OA}
  add PA[j] to {TA}
  NAtoms:=NAtoms + 1
  Atoms[NAtoms]:=PA[j]
  for Atoms[k] ∈ Molecules[j] do
  begin
   new AdjMatr[k1]
   AdjMatr[k1].i:=NAtoms
   AdjMatr[k1].j:=k
   AdjMatr[k1].m:=1
   looking for AdjMatr[k2].i=k
   AdjMatr[k2].j:=NAtoms
  end
 end
end
```

Both modes result in a simplified network array corresponding to the structural motif at a given crystal structure representation encoded by the collection ({OA}, {RA}, {CA}, {TA}) (*cf.* part 2.3). The net nodes are formed by the {OA} set; the resulted and initial nets are the same if {RA}={CA}={TA}=∅ in the **Atomic net** mode.

The network array may consist of several nets of the same or different dimensionality (0D–3D). Before topological classification *ADS* distinguishes all molecular (0D) groups, chain (1D), layer (2D) and framework (3D) nets in the array as is shown in the output for ODAHEG[7].

For each net *ADS* computes basic topological indices CS+ES+VS and several additional ones using original algorithms based on successful analysis of coordination shells. The algorithms have a number of advantages over described in the literature (Goetzke & Klein, 1991; O'Keeffe & Brese, 1992; Yuan & Cormack, 2002; Treacy *et al*., 2006):

- No distance matrix $D \times D$ is used, so the calculation is not memory-limited.
- There are no limits to the node degree (CN).
- Smallest ***circuits*** are computed along with smallest ***rings***.
- All rings, not only smallest, can be found within a specified ring size.
- ***Strong rings*** can be computed.

An example of *TOPOS* output with dimensionalities of structural groups in ODAHEG

```
###############################################################################
5;RefCode:ODAHEG:(C48 H60 CU1 N8 O6)N, 2N(C32 H42 CU1 N5 O4 +), 2N(N1 O3 -), N(C2 H6 O1)
Author(s): PLATER M.J.,FOREMAN M.R.ST.J.,GELBRICH T.,HURSTHOUSE M.B.
Journal: CRYSTAL ENGINEERING Year: 2001 Volume: 4 Number:  Pages: 319
```

[7] The CSD Reference Code.

```
################################################################################################
------------------------
Structural group analysis
------------------------

------------------------
Structural group No 1
------------------------
Structure consists of chains [ 0 1 0] with CuO6N8C48H56
2-c net

------------------------
Structural group No 2
------------------------
Structure consists of chains [ 0 0 1] with CuO4N5C32H42
2-c net
Elapsed time: 6.36 sec.
```

By computing all rings user can distinguish topologically different nets with the same CS+ES+VS combination. At present such examples are revealed only among artificial nets. The output has the following format:

An example of *TOPOS* output with all-ring Vertex symbols for rutile

```
Vertex symbols for selected sublattice
---------------------------------------
O1 Schlafli symbol:{4;6^2}
With circuits:[4.6(2).6(2)]
Rings coincide with circuits
All rings (up to 10): [(4,6(2)).(6(2),8(6)).(6(2),8(6))]
---------------------------------------
Ti1 Schlafli symbol:{4^2;6^10;8^3}
With circuits:[4.4.6.6.6.6.6.6.6.6.6(2).6(2).8(2).8(4).8(4)]
With rings:   [4.4.6.6.6.6.6.6.6.6.6(2).6(2).*.*.*]
All rings (up to 10):
[4.4.(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).6(2).6(2).*.*.*]
ATTENTION! Some rings * are bigger than 10, so likely no rings are contained in that angle
---------------------------------------
Total Schlafli symbol: {4;6^2}2{4^2;6^10;8^3}
```

In this case all rings were constructed up to 10-ring. So possibly larger rings exist - TOPOS does not know this!

The notation

```
All rings (up to 10): [(4,6(2)).(6(2),8(6)).(6(2),8(6))]
```

means that not only 4- (or 6-) rings, but also longer 8-rings meet at the same angle of the first non-equivalent node (oxygen atom, cf. ES or VS). There is still no conventional notation; it might look as: [(4,6_2).(6_2,8_6).(6_2,8_6)].

Resting upon the CS+ES+VS combination *ADS* searches for the net topological type in the *TTD* collection (part 2.2). Besides these basic indices, all rings and strong rings can be used for more detailed description of the net topology. A fragment of *ADS* output with the computed indices and the conclusion about the net topology is given below.

```
##################
63;RefCode:nbo:nbo
Author(s): Bowman A L,Wallace T C,Yarnell J L,Wenzel R G
Journal: Acta Crystallographica (1,1948-23,1967) Year: 1966 Volume: 21 Number:  Pages: 843
##################

Topology for C1
--------------------
Atom C1 links by bridge ligands and has
Common vertex with                              R(A-A)
C  1    0.0000    0.5000    0.0000   (-1 0 0)     1.000A        1
C  1    0.0000    0.5000    1.0000   (-1 0 1)     1.000A        1
```

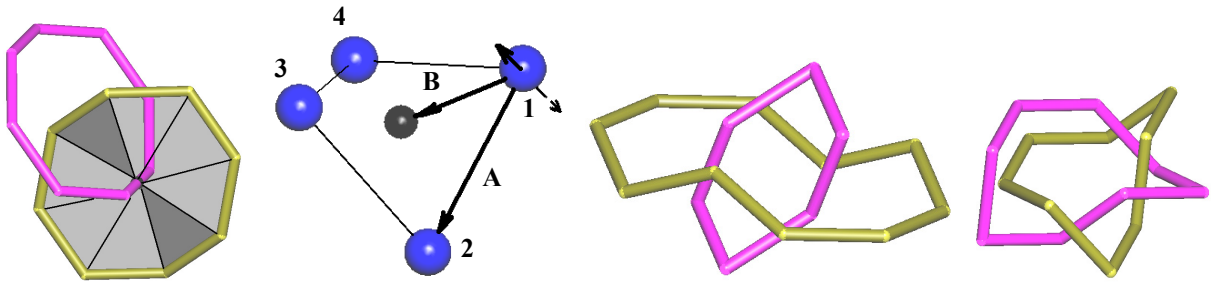
```
C  1    0.0000    0.0000    0.5000   ( 0-1 0)     1.000A        1
C  1    0.0000    1.0000    0.5000   ( 0 0 0)     1.000A        1

Coordination sequences
----------------------
C1:  1  2  3  4   5   6   7   8   9   10
Num  4 12 28 50  76 110 148 194 244  302
Cum  5 17 45 95 171 281 429 623 867 1169
----------------------
TD10=1169

Vertex symbols for selected sublattice
--------------------------------------
C1 Schlafli symbol:{6^4;8^2}
With circuits:[6(2).6(2).6(2).6(2).8(6).8(6)]
With rings:   [6(2).6(2).6(2).6(2).8(2).8(2)]
All rings (up to 10): [(6(2),8).(6(2),8).(6(2),8).(6(2),8).8(2).8(2)]
All rings with types: [(6(2),8).(6(2),8).(6(2),8).(6(2),8).8(2).8(2)]
--------------------------------------
Total Schlafli symbol: {6^4;8^2}
4-c net; uninodal net

Topological type: nbo NbO; 4/6/c2 {6^4;8^2} - VS [6(2).6(2).6(2).6(2).8(2).8(2)]  (18802 types in 6 data-
bases)

Strong rings (MaxSum=6): 6
Non-strong ring: 8=6+6+6+6
Elapsed time: 1.00 sec.
```



a *b* *c* *d*

Figure 8: *(a) Intersecting 8-rings (Hopf link) in self-catenating coesite; one of the rings is triangulated. (b) Two orientations (positive and negative) of the same 4-ring in body-centred cubic lattice determined as cross-products* ***A***×***B*** *and* ***B***×***A***. *The black ball is the ring centroid. The direction of the ring tracing (1234) coincides with the* ***A*** *direction. (c) Non-Hopf link between 6- and 10-ring in self-catenating ice II. (d) Double link between 8-rings in interpenetrating array of two quartz-like nets.*

If there are more than one nets in the array *ADS* determines the type of their mutual entanglement (*polythreading*, *polycatenation*, *interpenetration* and *self-catenation*) according to principles described by Carlucci *et al*. (2003), Blatov *et al*. (2004). Analysis of 0D–2D (low-dimensional) entanglements is based on searching for the intersections of rings by bonds not belonging to these rings. Since, generally speaking, the rings are not flat, they are represented as a facet surface by a barycentric subdivision (triangulation, Fig. 8*a*). The ring surface has two opposite orientations (positive and negative), and the ring boundary has a distinct direction of tracing (Fig. 8*b*). Let us call the ring intersection *positive* if the bond making an intersection within the boundary of the ring is directed to the same half-space as the vector of positive ring orientation, and *negative* otherwise. If there is the single ring intersection (positive or negative) the link between rings is always true (*Hopf*, Fig. 8*a*). If the numbers of positive and negative intersections are the same, the link can be unweaved (it is false, non-Hopf link, Fig. 8*c*), if the difference between the numbers is more than a unity, the link is multiple (Fig. 8*d*). *ADS* determines the link types; the real entanglement exists if there is at least one true (Hopf or multiple) link. Then *ADS* outputs the type of the entanglement (see Example 1). A special case is the entanglement of several 3D nets (3D interpenetration), when the information is output about Class of interpenetration (Blatov *et al*., 2004) and symmetry operations relating different 3D nets (Example 2).

Example 1. 2D+2D, inclined polycatenation (Fig. 9*a*)

```
########################################
6;RefCode:LETWAI:C24 H24 Cu4 F12 N12 Si2
```

Author(s): Macgillivray L.R.,Subramanian S.,Zaworotko M.J.
Journal: CHEM.COMMUN. Year: 1994 Volume: Number: Pages: 1325
##

Topology for Cu1

Atom Cu1 links by bridge ligands and has

Common vertex with					R(A-A)	f
Cu 1	0.7063	-0.2063	0.0000	(1 0 0)	6.937A	1
Cu 1	0.2063	0.2937	-0.5000	(0 0-1)	6.685A	1
Cu 1	0.2063	0.2937	0.5000	(0 0 0)	6.685A	1

Structural group analysis

Structural group No 1

Structure consists of layers (1 1 0); (1-1 0) with CuN3C6H6

Vertex symbols for selected sublattice

Cu1 Schlafli symbol:{6^3}
With circuits:[6.6.6]

Total Schlafli symbol: {6^3}
3-c net

Non-equivalent circuits

Circuit No 1; Type=6; Centroid: (0.500,0.000,0.500)

Atom	x	y	z
Cu1	0.2937	0.2063	1.0000
Cu1	0.7063	-0.2063	1.0000
Cu1	0.7937	-0.2937	0.5000
Cu1	0.7063	-0.2063	0.0000
Cu1	0.2937	0.2063	0.0000
Cu1	0.2063	0.2937	0.5000

Crossed with bonds

No	Atom	x	y	z	Atom	x	y	z	Dist.	N Cycles
1	Cu1	0.2937	-0.2063	0.5000	Cu1	0.7063	0.2063	0.5000	6.937	6/inf 6/inf

Ring links

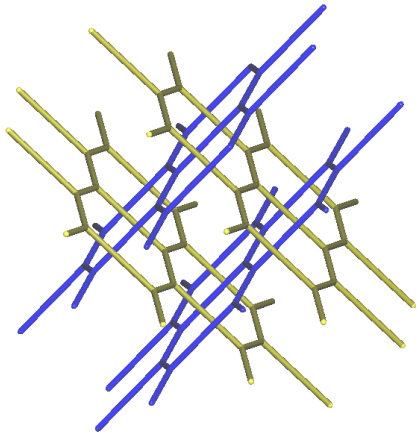
Cycle 1	Cycle 2	Chain	Cross	Link	Hopf	Mult
6	6	inf.	1	1	*	2

Polycatenation

Groups
1: 2D, CuN3C6H6 (Zt=1); (1,1,0); (1,-1,0)
Types

Group 1	Orient.	Group 2	Orient.	Type
1	1,1,0	1	1,-1,0	2D+2D, inclined

Elapsed time: 2.14 sec.



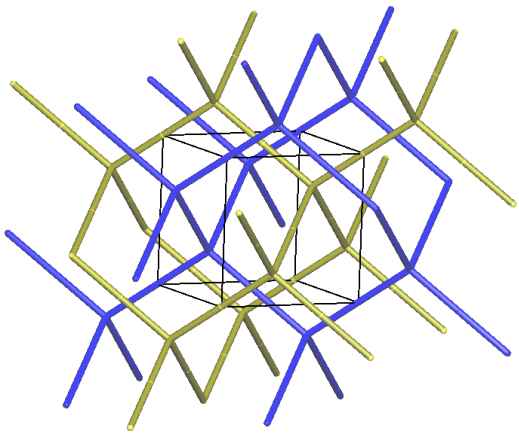


Figure 9: *(a) Entangled 2D layers in the crystal structure of LETWAI. The nets are simplified at {OA}={TA}={Cu}. (b) Interpenetrating 3D nets in the cuprite (Cu_2O) crystal structure.*

Example 2. Interpenetration of two 3D nets in cuprite, Cu_2O (Fig. 9*b*)

```
####################
7;RefCode:63281:Cu2O
Author(s): Restori R,Schwarzenbach D
Journal: Acta Crystallographica B (39,1983-) Year: 1986 Volume: 42 Number:  Pages: 201-208
####################

-------------------------
Structural group analysis
-------------------------

-------------------------
Structural group No 1
-------------------------
Structure consists of 3D framework with Cu2O
There are 2 interpenetrated nets
FIV: Full interpenetration vectors
---------------------------------
[0,1,0] (4.27A)
[0,0,1] (4.27A)
[1,0,0] (4.27A)
---------------------------------
PIC: [0,2,0][0,1,1][1,1,0] (PICVR=2)

Zt=2; Zn=1

Class Ia  Z=2

Vertex symbols for selected sublattice
--------------------------------------
O1 Schlafli symbol:{12^6}
With circuits:[12(2).12(2).12(2).12(2).12(2).12(2)]
--------------------------------------
Cu1 Schlafli symbol:{12}
With circuits:[12(6)]
--------------------------------------
Total Schlafli symbol: {12^6}{12}2
2,4-c net with stoichiometry (2-c)2(4-c)

-----------------------
Non-equivalent circuits
-----------------------
Circuit No 1; Type=12; Centroid: (0.000,0.500,0.500)
----------------------------
Atom       x       y       z
----------------------------
O1       0.2500  0.2500  1.2500
Cu1      0.5000  0.5000  1.0000
O1       0.7500  0.7500  0.7500
Cu1      0.5000  1.0000  0.5000
O1       0.2500  1.2500  0.2500
Cu1      0.0000  1.0000  0.0000
O1      -0.2500  0.7500 -0.2500
Cu1     -0.5000  0.5000  0.0000
O1      -0.7500  0.2500  0.2500
Cu1     -0.5000  0.0000  0.5000
O1      -0.2500 -0.2500  0.7500
Cu1      0.0000  0.0000  1.0000
Crossed with bonds
-------------------------------------------------------------------------------------------------------
  No | Atom       x       y       z     | Atom       x       y       z     |   Dist.  | N Cycles
-------------------------------------------------------------------------------------------------------
   1 | O1       -0.2500  0.7500  0.7500 | Cu1       0.0000  0.5000  0.5000 |    1.848 | 12/inf 12/inf
12/inf 12/inf 12/inf 12/inf
   1 | O1        0.2500  0.2500  0.2500 | Cu1       0.0000  0.5000  0.5000 |    1.848 | 12/inf 12/inf
12/inf 12/inf 12/inf 12/inf
-------------------------------------------------------------------------------------------------------

Ring links
----------------------------------------------------
Cycle 1 | Cycle 2 | Chain | Cross | Link | Hopf | Mult
----------------------------------------------------
     12 |      12 |  inf. |     1 |    1 |   *  |    6
----------------------------------------------------
```

ADS uses the information about ring intersections to construct ***natural tiling*** (Delgado-Friedrichs & O'Keeffe, 2005) that carries the net. Although the definition for natural tiling has been well known, there was no strict algorithm of its construction. The main problem is that not all strong rings (Fig. 10*a*) are necessary the faces of the tiles, but only *essential* ones (Delgado-Friedrichs & O'Keeffe, 2005; Fig. 10*b*). At the same time no criteria were reported to distinguish essential strong rings, so they can be determined only *after* constructing the natural tiling.

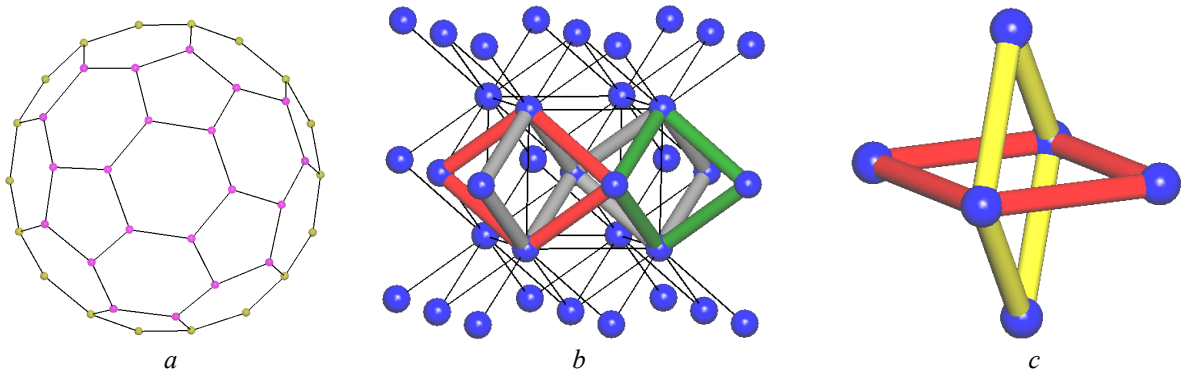


Figure 10: *(a) Closed sum of strong 5,6-rings (magenta) and non-strong 18-ring (yellow) in fullerene. (b) Two tiles, essential (green) and inessential (red) strong rings in the natural tiling of body-centred cubic net. (c) Two intersecting equivalent inessential rings (red and yellow) in the tile.*

ADS uses the following definition of essential strong ring: this is strong ring that intersects no other essential strong rings. There are two types of such intersections: *homo*crossing and *hetero*crossing, when the intersecting rings are equivalent (Fig. 10*c*) or inequivalent. The rings participating in a homocrossing are always inessential, the rings participating in only heterocrossings can be essential in an appropriate ring set, otherwise the ring is always essential. Thus, the algorithm of searching for essential rings consists of the following steps:

(i) Compute all rings within a given range. Because even the rings of the same size are not always symmetrically equivalent, *TOPOS* can distinguish them by assigning types. The types are designated by one or more letters: *a-z*, *aa-az*, *ba-bz*, *etc*., for example, 4*a*, 12*ab*, 20*xaz*. As a result a *typed* all-ring Vertex symbol is calculated:

An example of *TOPOS* output with typed all-ring Vertex symbols for rutile

```
Vertex symbols for selected sublattice
--------------------------------------
O1 Schlafli symbol:{4;6^2}
With circuits:[4.6(2).6(2)]
Rings coincide with circuits
All rings (up to 10): [(4,6(2)).(6(2),8(6)).(6(2),8(6))]
All rings with types: [(4,6(2)).(6(2),8a(4),8b(2)).(6(2),8a(4),8b(2))]
--------------------------------------
Ti1 Schlafli symbol:{4^2;6^10;8^3}
With circuits:[4.4.6.6.6.6.6.6.6.6.6(2).6(2).8(2).8(4).8(4)]
With rings:   [4.4.6.6.6.6.6.6.6.6.6(2).6(2).*.*.*]
All rings (up to 10):
[4.4.(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).(6,8(3)).6(2).6(2).*.*.*]
All rings with types:
[4.4.(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),8b).(6,8a(2),
 8b).6(2).6(2).*.*.*]
ATTENTION! Some rings * are bigger than 10, so likely no rings are contained in that angle
--------------------------------------
Total Schlafli symbol: {4;6^2}2{4^2;6^10;8^3}
```

For example, the first angle for the first node (oxygen atom) contains two non-equivalent 8-rings. There is no conventional notation for typed all-ring Vertex symbol. We propose the following one: [(4,6_2).(6_2,8a_4,8b_2).(6_2,8a_4,8b_2)].

(ii) Select strong rings. All non-strong rings are output as sums of smaller rings:

An example of *TOPOS* output with strong and non-strong rings for zeolite MTF

```
Strong rings (MaxSum=8): 4,5a,5b,5c,5d,6a,6b,6c,6d,8a,8b
Non-strong ring: 7=5d+6c+6d
Non-strong ring: 12=4+5a+5a+5b+5b+6a+8a+8b
```

(iii) Find all rings intersected by bonds (in entangled structures) and reject them. This condition is required because tile interior must be empty.

(iv) For all remaining rings find their intersections.

(v) Reject all rings participating in homocrossings.

(vi) Arrange all remaining rings into the sets, where no intersecting rings exist. The sets are *maximal*, *i.e.* no other ring can be added to the set to avoid heterocrossings.

Each of the sets obtained is then checked to produce a natural tiling. Starting from the first ring of the set and taking one of two possible ring orientations (Fig. 8*b*) *ADS* adds another ring to an edge of the initial ring to get a ring sum. For instance, three pentagonal and three hexagonal rings can be added to the central hexagonal ring in Fig. 10*a*. In 3D nets several (at least three) rings are adjacent to any edge, so there is an ambiguity at this step. To get over this problem and to speed up the calculation the dihedral angles are computed between each of the trial rings and the initial ring. Really these are the angles between normals to ring facets (triangles) based on the edge of the initial ring (Fig. 11*a*). Since the facets are oriented, the angles vary in the range 0-360°. Let us consider two facets of two trial rings 1 and 2 (candidates to be the tile face) with different angles φ_1 and φ_2; $\varphi_2>\varphi_1$. Obviously, if we choose the ring 2, this means that the tile intersects another tile to which the ring 1 belongs. So, the target ring for natural tile can be unambiguously chosen at each step as the ring with minimal dihedral angle φ_{min}. Then the next ring is added to any of *free*, *i.e.* belonging to only one ring, edge of the sum. The procedure repeats until no free edges remain, *i.e.* sum becomes *closed* (Fig. 10*a*). The closed ring sum is one of the natural tiles. Then the procedure starts again for the opposite orientation of the initial ring. As a result the initial ring becomes shared between two natural tiles (Fig. 11*b*). Then *ADS* considers all other inequivalent rings in the same way. Thus, all tiles forming the natural tiling are obtained with the following algorithm.

```
procedure Natural_Tiling(output Tiles)
NumTiles:=0
for i:=1 to NStrongRings do
begin
 NumTiles:=NumTiles + 1
 add StrongRings[i] to Tiles[NumTiles] initialize new tile
 for j:=1 to 2 do j is an orientation number for the first ring of the tile
 begin
  repeat
   call AddRing(j, output Tiles[NumTiles]) add new ring to the tile
  until no new ring is added to Tiles[NumTiles]
 end
end
```

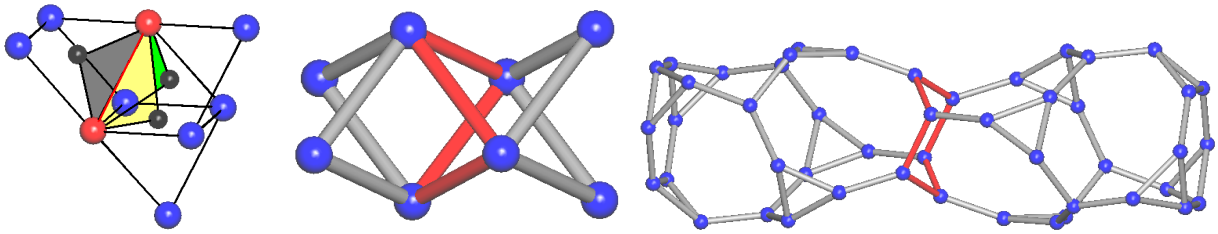


Figure 11: *(a) Some 4-rings sharing the same (red) edge in body-centred cubic net. The grey facet is the facet of the initial ring; the yellow one has smaller φ than the green one. The black balls are the rings centroids. (b) 4-ring (red) shared between two natural tiles. (c) Two natural tiles shared by red face in an MPT of the idealized net* ***bcw****.*

Then *ADS* determines a number of geometrical and topological characteristics of tiles and tiling (Delgado-Friedrichs & O'Keeffe, 2005). The resulted output looks as shown below (the sodalite net example).

The physical meaning of the tiles is that they correspond to minimal cages in the net. Using these 'bricks' *ADS* can construct larger tiles by summarizing natural tiles (merging them by faces). In this way, *maximal* proper tiles (MPT) and tiling can be obtained representing maximal cages allowed by a given net symmetry (Fig. 11*c*).

Resting upon the tiling *ADS* can construct *dual* net, whose nodes, edges, rings and tiles map onto tiles, rings, edges and nodes of the initial net (Delgado-Friedrichs & O'Keeffe, 2005). In particular, nodes and edges of the dual net describe the topology of the system of cages and channels in the initial net (Fig. 12). The data on the dual net are stored in a *TOPOS* database, so the dual net can be studied as an ordinal net including generation of dual net ('dual dual net').

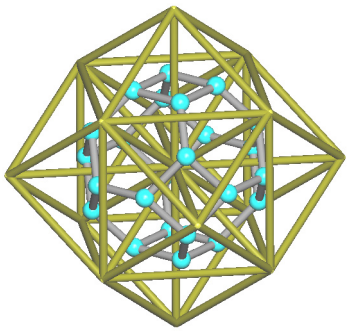


Figure 12: *Initial net (cyan balls) and dual net (yellow sticks) in sodalite.*

An example of *TOPOS* output for natural tiling in sodalite net

```
#################
3;RefCode:sod:sod
#################

Topology for C1
-------------------
Atom C1 links by bridge ligands and has
Common vertex with                                  R(A-A)
C  1    0.5000    0.0000    0.2500   ( 1 0 0)     0.707A        1
C  1    0.5000    0.0000    0.7500   ( 1 0 1)     0.707A        1
C  1    0.0000    0.2500    0.5000   ( 0 0 1)     0.707A        1
C  1    0.0000   -0.2500    0.5000   ( 0 0 1)     0.707A        1

Vertex symbols for selected sublattice
--------------------------------------
C1 Schlafli symbol:{4^2;6^4}
With circuits:[4.4.6.6.6.6]
Rings coincide with circuits
Rings with types: [4.4.6.6.6.6]
--------------------------------------
Total Schlafli symbol: {4^2;6^4}
4-c net

Essential rings by homocrossing: 4,6
Inessential rings by homocrossing: none
-----------------------------
Primitive proper tiling No 1
-----------------------------

Essential rings by heterocrossing: 4,6
Inessential rings by heterocrossing: none

Natural tiling
24/14:[4^6.6^8]; Centroid:(0.500,0.500,0.500); Volume=4.000; G3=0.078543
--------------------------------
Atom       x       y       z
--------------------------------
C1       0.5000  0.2500  0.0000
C1       0.5000  0.0000  0.2500
C1       0.7500  0.0000  0.5000
C1       0.2500  0.0000  0.5000
C1       0.5000  0.0000  0.7500
C1       0.5000  0.2500  1.0000
C1       0.0000  0.2500  0.5000
C1       0.2500  0.5000  0.0000
C1       0.0000  0.5000  0.2500
C1       0.7500  0.5000  0.0000
C1       0.5000  0.7500  0.0000
C1       1.0000  0.2500  0.5000
C1       1.0000  0.5000  0.2500
C1       0.7500  0.5000  1.0000
C1       1.0000  0.5000  0.7500
C1       1.0000  0.7500  0.5000
C1       0.0000  0.7500  0.5000
C1       0.0000  0.5000  0.7500
C1       0.2500  0.5000  1.0000
C1       0.5000  0.7500  1.0000
C1       0.5000  1.0000  0.2500
C1       0.7500  1.0000  0.5000
C1       0.5000  1.0000  0.7500
C1       0.2500  1.0000  0.5000

Tiling: [4^6.6^8]
Transitivity: [1121]
Simple tiling

All proper tilings (S=simple; I=isohedral)
-------------------------------------------------------------
Tiling   | Essential rings | Transitivity | Comments | Tiles
-------------------------------------------------------------
PPT 1/NT | 4,6             | [1121]       | MPT SI   | [4^6.6^8]
-------------------------------------------------------------

Elapsed time: 2.55 sec.
```

3.2.2. Analysis of systems of cavities and channels

Quite another way to get the system of cages and channels is to consider Voronoi-Dirichlet partition (part 2.1) and to analyze the net of VDP vertices and edges, ***Voronoi-Dirichlet graph*** (Fischer, 1986). The principal difference between tiling and Voronoi-Dirichlet approaches is that the former approach is purely topological and derives the cages and channels from the *topological* properties of the initial net, whereas the latter one treats the *geometrical* properties of crystal space for the same purpose. Here the geometrical and topological parts of *TOPOS* are combined with each other.

The main notions of the Voronoi-Dirichlet approach are ***elementary void*** and ***elementary channel***. Some their important properties to be used in *TOPOS* algorithms follow from the properties of Voronoi-Dirichlet partition.

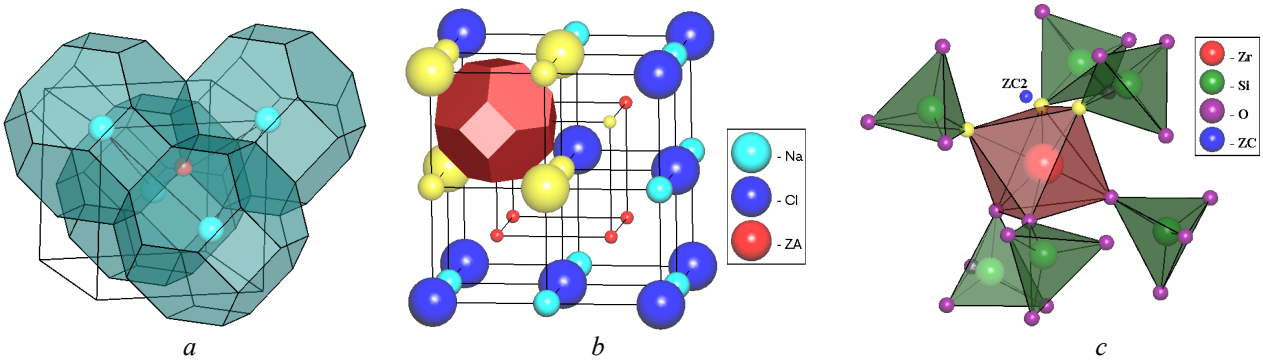
Elementary void properties

(i) The elementary void is equidistant to at least four noncoplanar atoms (tetrahedral void) since no less than four VDPs meet in the same vertex (Fig.13*a*). There are two types of elementary voids: *major*, if its centre is allocated inside the polyhedron, whose vertices coincide with the atoms forming the elementary void (for instance, inside the tetrahedron for a tetrahedral void, Fig.13*a*); and *minor*, if its centre lies outside or on the boundary of the polyhedron (Fig.13*b*).

(ii) There are additional atoms at longer distances than the atoms of elementary void that can strongly influence the geometrical parameters of the elementary void. To find these parameters, one should construct the void VDP taking into account all atoms and other equivalent elementary voids (Fig.13*c*). Let us call the atoms and voids participating in the VDP formation *environmental*. Obviously, the atoms forming the elementary void are always environmental.

(iii) *Radius* of elementary void (R_{sd}) is the radius of a sphere, whose volume is equal to the volume of the void VDP constructed with consideration of all environmental atoms and voids.

(iv) *Shape* of elementary void is estimated by G_3 value for the void VDP constructed with all environmental atoms and voids (Fig.13*c*).



a *b* *c*

Figure 13: (*a*) Four VDPs meeting in the same vertex (red ball) in the body-centred cubic lattice. (*b*) A minor void (ZC2) allocated outside the tetrahedron of the three yellow oxygen atoms and zirconium atom forming this void in the crystal structure of NASICON, $Na_4Zr_2(SiO_4)_3$. All distances from ZC2 to the oxygen and zirconium atoms are equal 1.722 Å. (*c*) The form of an elementary void in the NaCl crystal structure. All environmental atoms and one void are yellow; R_{sd} =1.38 Å, G_3=0.07854.

Elementary channel properties

(i) The elementary channel is formed by at least three noncollinear atoms since in the Voronoi-Dirichlet partition each VDP edge is shared by no less than three VDPs. The plane passing through these atoms is perpendicular to the line of the elementary channel (Fig. 14*a*).
(ii) *Section* of the elementary channel is a polygon whose vertices are the atoms forming the channel; the section always corresponds to the narrowest part of the channel. The line of the elementary channel is always perpendicular to its section; ordinarily, the channel section and channel itself are triangular (Figs. 14*a*, *b*). The elementary channel can be of two types: *major*, if its line intersects its section (Fig. 14*a*), and *minor*, if the line and section have no common points, or one of the line ends lies on the section (Fig. 14*b*).

(iii) *Radius* of the elementary channel section is estimated as a geometric mean for the distances from the inertia centre of the elementary channel section to the atoms forming the channel. The atom can freely pass through the channel if the sum of its radius and an averaged radius of the atoms forming the channel does not exceed the channel radius.

(iv) *Length* of elementary channel is a distance between the elementary voids connected by the channel, *i.e.* is the length of corresponding VDP edge.

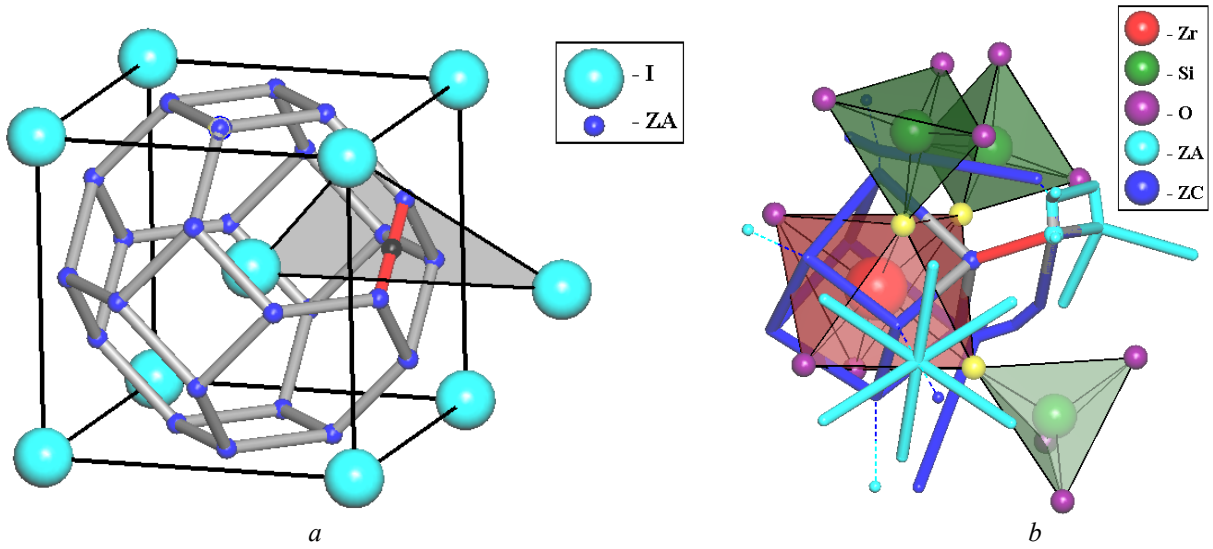


Figure 14: (*a*) Section of a triangular major elementary channel in the crystal structure of α-AgI. The channel line is red. The atoms forming the channel are in the vertices of the triangular section intersecting the channel line in the black ball. (*b*) A fragment of the channel system in the crystal structure of NASICON. The line of a minor elementary channel is red, the oxygen atoms forming this channel are yellow. Other minor elementary channels are shown by dotted lines.

The Voronoi-Dirichlet approach is implemented into the program *Dirichlet* as the following general algorithm:

(i) constructing VDPs for all independent framework atoms, *i.e.* a Voronoi-Dirichlet partition of the crystal space (interstitial particles including mobile ions or solvate molecules are ignored);

(ii) determining the coordinates for all independent vertices of atomic VDPs, and, as a result, the coordinates of all elementary voids;

(iii) determining all independent VDP edges, and, hence, all elementary channels;

(iv) calculating the numerical parameters of elementary voids and channels.

The information on the resulted conduction pattern is stored as a three-level adjacency matrix of the Voronoi-Dirichlet graph (Fig. 15).

The *first* level contains the information on a (central) elementary void. A major elementary void is designated as ZA, a minor one is marked as ZB or ZC if it lies respectively on the boundary or outside the polyhedron of the atoms forming the void. The void radius (R_{sd}, Å) and second moment of inertia of its VDP (G_3) are also shown.

The *second* level includes the information on other elementary voids connected with a given (central) one by channels, and on the atoms of its near environment. Every elementary void is characterized by the length (**R**, Å) of the elementary channel connecting it with the central one, by the number of channel atoms (**Chan**), and by the channel radius (**Rad**, Å). If a channel is major, the text is marked bold, otherwise a normal font is applied. Every environmental atom of the central void is characterized by the distance to the void centre (**R**, Å), and by the solid angle of corresponding VDP face (**SA**, in percentage of 4π steradian); the greater SA, the more significant the contact atom–void.

The *third* level contains the information on the atoms forming the channel; the distances between the atoms and the centre of the channel section are also given.

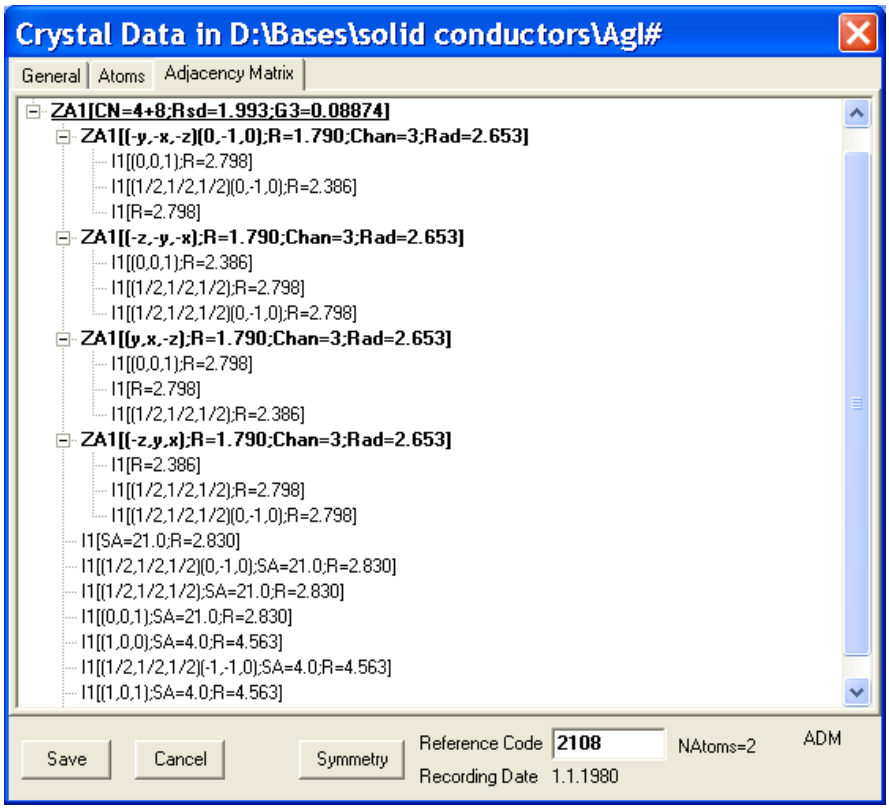


Figure 15: *A TOPOS window containing the information on the adjacency matrix of the Voronoi-Dirichlet graph for α-AgI.*

In contrast to the tiling approach, resting upon the adjacency matrix of Voronoi-Dirichlet graph *TOPOS* can compute a number of geometrical parameters of cages and channels to be important to predict some physical properties of the substance, in particular, ionic conductivity and ion-exchange capacity. Note that, in general, the Voronoi-Dirichlet graph does not coincide with the dual net. The dual net has always the same topology for any spatial embedding of the initial net, whereas the topology of the Voronoi-Dirichlet graph depends on the geometrical properties of the crystal space. The physical meanings of the Voronoi-Dirichlet graph parameters are summarized in Table 3; recall that the graph nodes and edges correspond to elementary voids and channels.

Table 3: *Physical meaning of Voronoi-Dirichlet graph parameters*

Parameter	Meaning
R_{sd} of node	The radius of an atom that can be allocated in the void under the influence of the crystal field distorting the spherical shape of the atom
G_3 of node	Sphericity degree for the nearest environment of the void; the less G_3, the closer the void shape to a sphere
Radius of edge	Effective radius of the channel between two voids
Length of edge	Length of the channel between two voids
Connected subgraph	Migration path, a set of the elementary voids and channels available for mobile particles
A set of all connected subgraphs	Conduction pattern of the substance. The dimensionality of the conduction pattern determines the dimensionality of conductivity (1D, 2D, or 3D)

3.2.3. Analysis of packings

Besides the model of atomic net, *TOPOS* can use the structure representation as a packing of atoms or atomic groups. In this case the adjacency matrix of the crystal structure contains interatomic *contacts*, not *bonds*. This representation can be obtained running *AutoCN* in the **Solid Angles** mode (part 2.1). The program *IsoTest* internally uses this mode to enumerate all possible atom packings to be selected in the crystal structure. *IsoTest* forms all subsets of the {NAtoms} set of all kinds of atom and generates the *packing* net (*cf*. part 2.3) by considering all faces of atomic VDPs constructed within a given subset. For instance, for NaCl, there will be considered three ion packings corresponding to the subsets {Na}, {Cl} and {Na, Cl}. For each of them *IsoTest* computes topological indices and performs topological analysis according to Scheme 4.

3.2.4. Hierarchical topological analysis

At the highest level of the topological analysis (Scheme 4) *IsoTest* analyzes all graph representations with the algorithm described in part 2.3. Simultaneously, *IsoTest* arranges the compounds by structure types using the definition of Lima-de-Faria *et al*. (1990). The results are output into a textual `*.it2` file as shown below for comparing simple sulfates with binary compounds.

```
------------------
Isotypic compounds
----------------------------
Topological type of 1:Li2(SO4)
----------------------------
(SO4)+Li
 3:CaF2: F+Ca
----------------------------
Topological type of 6:CaSO4
----------------------------
(SO4)+Ca
 1:NaCl: Na+Cl
----------------------------
Topological type of 7:ZnSO4
----------------------------
   Structure Type of 34:SiO2 S+Zn<->Si O<->O
Zn+(SO4)
 5:ZnS: Zn+S
----------------------------
Topological type of 17:MgSO4
----------------------------
(SO4)+Mg
 2:NiAs: Ni+As
```

In particular, the results show that Li_2SO_4 relates to the fluorite, CaF_2, if sulfate ion is considered as a whole, with the oxygen atoms contracted to the sulfur atom. The similar relations are observed for the pairs $CaSO_4 \leftrightarrow NaCl$; $ZnSO_4 \leftrightarrow ZnS$; $MgSO_4 \leftrightarrow NiAs$. However, $ZnSO_4$ has one more relation to cristobalite, SiO_2, if Zn and S atoms correspond to Si atoms.

Thus, the general scheme starts with the analysis of a single net or packing consisting of atoms, ions, molecules, voids, and finishes by the consideration of all possible topological motifs.

4. Processing large amounts of crystal structure data

Most of *TOPOS* procedures and applied programs can work in two modes, **Manual** or **Continuous**, corresponding to handling the single compound or large groups of crystal structures, respectively. The only exception is *IsoCryst*, where the **Continuous** mode is not available. The **Continuous** mode is not restricted to the number of entries; the largest world-wide databases, CSD, ICSD, CrystMet, may be processed at one computational cycle using the CIF interface. The data obtained in the **Continuous** mode are output to external files to be handled with *TOPOS* or other programs. The main **Continuous** operations available in *TOPOS* are listed below.

Operation	Output file format
DBMS	
Copying, moving, deleting, undeleting, searching, retrieving, exporting, importing database entries	*TOPOS* database, textual files
Determining chemical composition, searching for errors in data and disordering, transforming adjacency matrix, generating crystal structure representations	*TOPOS* database
ADS	
Simplifying atomic net	*TOPOS* database
Computing CS, ES, VS, determining net topology	textual `*.nnt`, Microsoft Excel-oriented `*.txt`
Determining net entanglements and structure group dimensionality	Microsoft Excel-oriented `*.txt`
Selecting molecular crystal structure groups, constructing molecular VDPs, determining methods of ligand coordination	binary *StatPack* `*.bin`
Constructing tiles, computing parameters of natural tiling	textual `*.cgd`
Determining combinatorial types of tile	binary `*.edg`, `*.pdt`, `*.vec`
Constructing dual net	*TOPOS* database
AutoCN	
Computing adjacency matrix	*TOPOS* database
DiAn	
Computing interatomic distances and bond angles	textual `*.dia`, `*.ang`
Dirichlet	
Computing atomic VDP parameters	binary *StatPack* `*.bin`
Determining combinatorial types of VDP	binary `*.edg`, `*.pdt`, `*.vec`
Constructing Voronoi-Dirichlet graph	*TOPOS* database
HSite	
Determining positions of hydrogen atoms	*TOPOS* database
IsoTest	
Generating crystal structure representations, computing CS, ES, VS	*TOPOS* database
Comparing atomic and packing net topologies	textual `*.it2`
Determining structure types	textual `*.ist`

5. Outlook

Crystallochemical analysis is still mainly based on the analysis of *local* geometrical properties of crystal structures, such as interatomic distances and bond angles. In this manner, *crystal* chemistry remains to be *stereo*chemistry to a great extent. Discovering genuine *crystallo*chemical regularities that manage global properties of crystal structures, such as atomic and molecular net topologies, types of atomic and molecular packings, sizes and architecture of cages and channels, require new theoretical approaches, computer algorithms and programs. To find these regularities crystal chemists need first to systematize huge amounts of crystal data collected in the world-wide electronic databases. This job can be done only by using automated computer methods, and *TOPOS* is intended to actualize them. The transformation of crystal *chemistry* into *crystal* chemistry notably began 15-20 years ago, but has already given rise to novel scientific branches, such as supramolecular chemistry, reticular chemistry, crystal engineering and crystal design. *TOPOS* program package evolves together with crystal chemistry; new algorithms and procedures are implemented every year. In this way, the current *TOPOS* state described above should not be considered as something completed, but as a foundation for further development.

Acknowledgements

The *TOPOS* applied programs *Dirichlet*, *IsoCryst* and *StatPack* were mainly written by Dr. A.P. Shevchenko. I am indebted to my former scientific advisor Prof. V.N. Serezhkin, who initialized the *TOPOS* project at the end of 1980s and stimulated its development for a long time. I am grateful to Prof. D.M. Proserpio, who opened my eyes to a lot of crystallochemical problems to apply *TOPOS*. Discussions with Prof. M. O'Keeffe, Dr. S.T. Hyde, Dr. O. Delgado-Friedrichs have highly promoted the development of *TOPOS* topological algorithms. My PhD students I.A. Baburin, E.V. Peresypkina, M.V. Peskov spent a lot of time testing novel *TOPOS* features; their painstaking work enabled me to fix many bugs and provided high *TOPOS* stability.

References

Blatov, V. A. (2004). *Cryst. Rev.* **10**, 249-318.
Blatov, V. A. (2006). *Acta Cryst*. **A62**, 356-364.
Blatov, V. A., Carlucci, L., Ciani, G. & Proserpio, D. M. (2004). *CrystEngComm*, **6**, 377–395.
Carlucci, L., Ciani, G. & Proserpio, D. M. (2003). *Coord. Chem. Rev.* **246**, 247–289.
Chung, S. J., Hahn, Th. & Klee, W. E. (1984). *Acta Cryst*. **A40**, 42-50.
Conway, J.H. & Sloane, N.J.A. (1988). *Sphere Packings, Lattices and Groups*, New York: Springer Verlag.
Delgado-Friedrichs, O. & O'Keeffe, M. (2005). *J. Solid State Chem*. **178**, 2480-2485.
Fischer, W. (1986). *Cryst. Res. Technol*. **21**, 499-503.
Goetzke, K. & Klein, H.-J. (1991). *J. Non-Cryst. Solids*. **127**, 215-220.
Lima-de-Faria, J., Hellner, E., Liebau, F., Makovicky, E. & Parthé, E. (1990). *Acta Cryst*. **A46**, 1-11.
O'Keeffe, M. (1979) *Acta Cryst.* **A35**, 772-775.
O'Keeffe, M. & Brese, N.E. (1992). *Acta Cryst*. **A48**, 663-669.
Peresypkina, E.V. & Blatov, V.A. (2000). *Acta Cryst.* **B56**, 1035-1045.
Preparata, F. P. & Shamos, M. I. (1985). *Computational Geometry*. New York: Springer-Verlag.
Serezhkin, V. N., Mikhailov, Yu. N. & Buslaev, Yu. A. (1997). *Russ. J. Inorg. Chem*. **42**, 1871-1910.
Sowa, H. & Koch, E. (2005). *Acta Cryst*. **A61**, 331-342.
Treacy, M.M.J., Foster, M.D. & Randall, K.H. (2006). *Microp, Mes. Mater*. **87**, 255–260.
Yuan, X. & Cormack, A.N. (2002). *Comput. Mater. Sci.* **24**, 343-360.

Appendix - *TOPOS* Glossary

Atomic domain is a region, which 'belongs' to an atom in crystal space. The notion 'belongs' may be understood differently, depending on the task to be solved.

Atomic Voronoi-Dirichlet polyhedron (VDP, Voronoi polyhedron, Dirichlet domain) is a convex polyhedron whose faces are perpendicular to segments connecting the central atom of VDP (*VDP atom*) and other (*surrounding*) atoms; each face divides corresponding segment by half. VDPs of all atoms form normal (face-to-face) *Voronoi-Dirichlet partition* of crystal space.

Circuit (*cycle*) is a closed chain of connected atoms.

Coordination sequence (CS) $\{N_k\}$ is a set of sequential numbers N_1, N_2, … of atoms in 1st, 2nd, *etc*. coordination spheres of an atom in the net. The first ten coordination spheres are usually considered at the topological classification. The coordination number is equal to N_1, and the graph node is called *N_1-connected* or *N_1-coordinated*.

Elementary channel is a free space connecting a couple of *elementary voids*; the channel corresponds to a VDP edge for the atom forming either of the voids. Such an edge is called *line* of the elementary channel. Accordingly, the atoms *forming* the elementary channel are the atoms whose VDPs have common edge coinciding with the channel line.

Elementary void is a region of crystal space with the centre in a vertex of an atomic VDP. The atoms, whose VDPs meet in the centre of a given elementary void, are referred to as atoms *forming* the elementary void**.**

Extended Schläfli symbol (ES) contains a detailed description of all shortest *circuits* for each angle at each non-equivalent atom. *Total Schläfli symbol* summarizes all the Schläfli symbols for the non-equivalent atoms with stoichiometric coefficients.

Indirect neighbours are contacting atoms, the segment between which does not intersect the VDP face separating these atoms (*minor* VDP face). Otherwise atoms are called *direct neighbours* and the corresponding VDP face is called *major*.

Lattice quantizer is a multilattice embedded into space in such a way that any point of the space is rounded to the nearest node of the quantizer.

Molecular Voronoi-Dirichlet polyhedron is a union of VDPs of atoms composing a molecular (0D) structural group. *Facet* is a face of the VDP of an atom belonging to a molecular structure group. All facets corresponding to contacts between two molecules form *faces* (*boundary surfaces*) of adjacent molecular VDPs. *Smoothed molecular VDP* is a convex polyhedron derived from the molecular VDP by flattening their faces.

Natural tiling is such subdivision of space by *tiles* that (i) preserves the symmetry of the net, (ii) has strong rings as tile faces, (iii) contains the tiles as small as possible.

Radius of spherical domain (R_{sd}) is the radius of a sphere of VDP volume.

Ring is the *circuit* without shortcuts, *i.e.* chains between two ring nodes that are shorter than any chain between the nodes that belongs to the circuit.

Strong ring is the *ring* that is not a sum of smaller rings.

Tile is a 3D solid (generalized polyhedron) bounded by *rings* (faces) in such a way that any ring edge is shared between two rings.

Vertex symbol (VS) gives similar information as *extended Schläfli symbol*, but for *rings*.

Voronoi-Dirichlet graph is the graph consisting of all vertices and edges of all VDPs in the *Voronoi-Dirichlet partition*.

The XPac Program for Comparing Molecular Packing

Thomas Gelbrich
School of Chemistry, University of Southampton, Highfield, Southampton, SO17 1BJ, UK. E-mail: gelbrich@soton.ac.uk *; WWW:* http://www.ncs.chem.soton.ac.uk/thomas.htm

Introduction

The lack of suitable automated procedures has been a major obstacle in investigations of packing relationships in molecular crystals. Potential applications are not only the study of polymorphic forms, multiple-component crystals and close analogues of a given molecule, but also the comparison of sets of predicted structures. In a recent publication,[1] we have reported a method for the identification of similar packing components which is implemented in the program XPac. Its effectiveness was demonstrated in a number of case studies.[1] More recently, we have reported another application involving a larger series of 25 structures based on the carbamazepine molecule,[2] and more examples have been published elsewhere.[3–6] We found that packing similarities are commonplace across a wide range of structures. In the present paper, we will take a closer look at the techniques employed by XPac. We will begin with a brief summary of the basic ideas.

Concept

The unit cell parameters, the space group and a list of atomic coordinates provide data for a concise description of the arrangement of molecules in a crystal structure. Another, more expansive, possibility is the construction of the first coordination environment around each independent molecule. We call such an assembly of molecules a representative ***cluster*** of the crystal structure. It consists of a central molecule or ***kernel*** surrounded by *m* molecules forming a ***shell***. The spatial arrangement of molecules in a given cluster can be described with internal coordinates. Thus, the clusters of two crystal structures containing molecules of a similar shape and size are immediately comparable. This applies regardless of the crystal systems, space groups and Z' parameters (number of crystallographically independent molecules) involved.

Furthermore, a systematic comparison of the ***sub-units*** of two clusters will ultimately reveal if the underlying structures have any packing component in common. Such a component, termed a ***supramolecular construct*** (SC), is either extended in one, two or three dimensions or it is a discrete building block, and its representation in a cluster is called a ***seed***. Hence, we compare two crystal structures by comparing their representative clusters, enabling us to identify an SC via its seed.

Representative cluster

After obtaining the crystal structure parameters from a cif-file, all *Z′* representative clusters of a structure are computed by investigation of a block of 3 × 3 × 3 unit cells. A shell molecule contains at least one atom for which the condition

$$d_{k,s} < (r_k + r_s + d)$$

is satisfied ($d_{k,s}$ = distance between this atom and one atom belonging to the kernel molecule; r_k, r_s= van der Waals radii of these two atoms; d = cut-off parameter). The parameter d is adjustable and has a default value of 1.5 Å. The possible inclusion of unnecessary molecules does not compromise subsequent calculations. A structure with Z' = 2 contains two overlapping clusters. The following demonstration will be restricted to two crystal structures A and B with Z' = 1. The same rules apply also to Z' > 1 cases, though the treatment becomes more complex.

Corresponding Sets of Points

A set of points, derived from atomic positions, is assigned to the molecules of A and B, respectively. It will be used later for the generation of internal parameters. The positions chosen here will therefore determine the character of this investigation. Normally, one would aim to represent the shape of the molecule and avoid, if possible, arrangements that are coplanar or even collinear. However, for some investigations it is necessary to choose points that define just a particular region of the molecule.[1a]

The application of XPac is only sensible if it is possible to establish corresponding ordered sets of points (COSP) for the molecules of structures A and B. This means, each molecule is represented by an ordered set of N points (OSP), say $K = \{P_1, P_2, ...P_N\}$ for structure A and $\underline{K} = \{\underline{P}_1, \underline{P}_2, ...\underline{P}_N\}$ for structure B and there is a relation or correspondence, between each point P_i and each point $\underline{P}_i$ where $1 \le i \le N$.

Figure 1a shows a typical choice of six non-hydrogen positions. Note that this arrangement is not affected by rotation of the benzene rings about the N–C and the S–C bond, respectively. This makes it more likely that a geometrically consistent COSP will be generated. All subsequent test procedures rely on differences in corresponding internal parameters computed from the COSP. These arise from two main sources: from the actual difference in molecular packing, which is the desired information, and from any inconsistencies in the COSP, which should be minimised. We can obtain a rough measure for the consistency of a COSP by calculating the mean absolute difference between all corresponding angles $\angle(P_i, P_j, P_k)$ and $\angle(\underline{P}_i, \underline{P}_j, \underline{P}_k)$ with $1 \le i, j, k \le N$ and $i \ne j$, $i \ne k$, $j \ne k$.

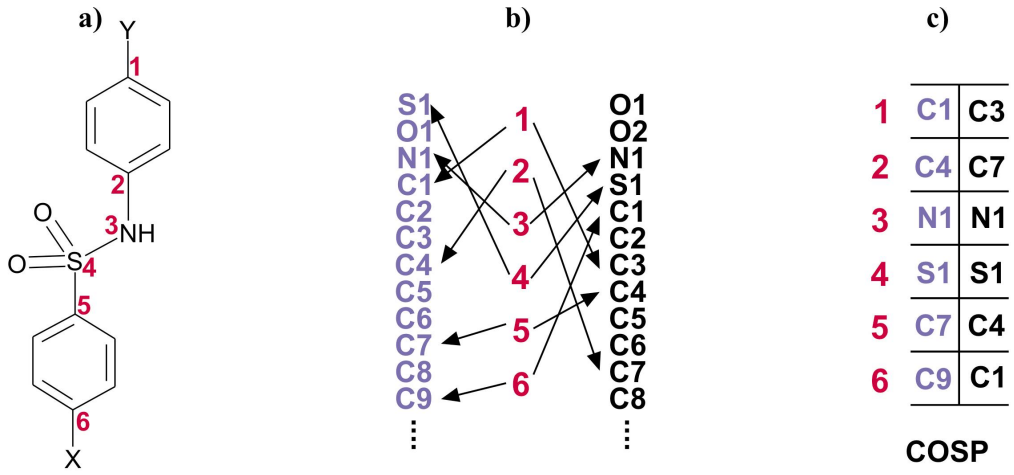


Fig. 1: *Corresponding points for two crystal structures. a) Scheme showing the order of points. b) Generation of two corresponding sets of points from the differently ordered atom list of structures A (left) and B (right). c) The resulting corresponding sets of points.*

Sub-units of Clusters

The clusters of structures A and B contain m and n shell molecules, respectively. The two cluster kernels are represented by sets of points, K and $\underline{K}$, as defined in the previous step. Now we compute the shell for the first structure by applying the appropriate symmetry operations on K so that m new OSP, S^1, S^2, ...S^m are obtained. The shell $\underline{S}^1$, $\underline{S}^2$, ...$\underline{S}^n$ of the cluster of B is computed in the same fashion. We have now obtained a representation for each complete cluster,

$$\mathbf{C} = \{K, S^1, S^2, ...S^m\} \quad \text{and} \quad \underline{\mathbf{C}} = \{\underline{K}, \underline{S}^1, \underline{S}^2, ...\underline{S}^n\}.$$

Next we consider subsets of **C** and $\underline{\mathbf{C}}$ that are composed of the kernel and at least one shell molecule. We have argued earlier that the presence of a similar packing arrangement leads to similar cluster fragments, i.e. two cluster subsets **E** and $\underline{\mathbf{E}}$ of **C** and $\underline{\mathbf{C}}$, respectively, containing a similar spatial arrangement of points (molecules). We use the "~" sign to indicate geometrical similarity, $\mathbf{E} \sim \underline{\mathbf{E}}$.

E is then the seed of a sub-cluster (SC) in cluster **C**, and **E** is its counterpart in **C**. Both contain x shell molecules (and the kernel) and each element of **E** corresponds to one particular element of **E** and vice versa. Furthermore, seeds that contain more than one shell molecule, i.e.

$\mathbf{E} = \{K, S_1, S_2 \ldots S_x\}$ and $\mathbf{E} = \{K, S_1, S_2 \ldots S_x\}$ where $1 < x \leq \min(m, n)$,

can be divided into $x^2 - x / 2$ overlapping ***triple sub-units*** (composed of two shell molecules and the kernel), and all these corresponding triple sub-units must be similar,

$\{K, S_i, S_j\} \sim \{ K, S_i, S_j \}$ for all i, j with $1 \leq i, j \leq x$ and $i \neq j$.

Furthermore, we base our strategy on the assumption that, in order to show that **E** ~ **E**, it is sufficient to show that this condition is satisfied for *all* pairs of corresponding constituent triple sub-units. Thus, we will first establish whether **C** and **C** contain any similar ***double sub-units***, {K, S} ~ {K, S}. Then, the SC will be built gradually from such pairs of double sub-units via triple sub-units and the seed.

First, we divide **C** and **C** into m and n double sub-units, {K, S} and {K, S}, respectively, each consisting of the kernel and one shell molecule, as illustrated in Fig. 2b. These are also the smallest packing fragments that two structures can have in common. We use three different types of parameter for the parameterisation of the spatial arrangement in {K, S}. The first type, *ang*, is the angle ∠(A, B, C) where either (A, B ∈ K; C ∈ S) or (A ∈ K; B, C ∈ S). The second type, *dhd*, is computed from five points A, B, C, D, E. It is the angle between the normal directions of the two planes defined by points (A, B, C) and (C, D, E), respectively, where (A, B, C ∈ K and D, E ∈ S) or (A, B ∈ K and C, D, E ∈ S). The third type, *tor*, is the torsion angle obtained from four points A, B, C, D. It is the complement of the angle between two planes defined by (A, B, C) and (B, C, D) where (A, B, C ∈ K and D ∈ S) or (A, B ∈ K and C, D ∈ S) or (A ∈ K and B, C, D ∈ S). The use of other parameters or combinations of parameters is possible, distance parameters are not employed in the current version of the program.

A list of parameters, which is divided into three sections (*ang*, *dhd* and *tor*) and has a sufficiently large number of entries is compiled in exactly the same fashion for each {K, S} and each {K, S} sub-unit. Thus, all list entries with the same index correspond to one another.

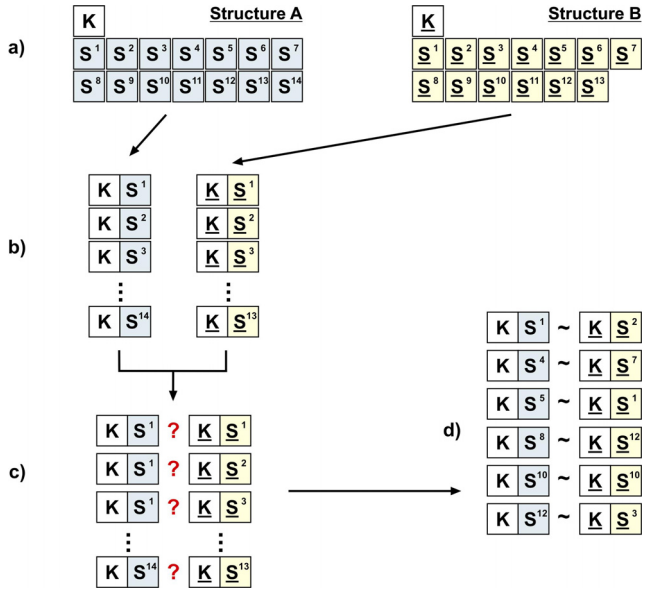


Fig. 2: *a) Representation of the clusters of two structures. The cluster **C** of structure A consists of the kernel molecule and 14 shell molecules. The cluster **C** of structure B contains 13 shell molecules. b) Generation of double sub-units {K, D} and {K, S}. c) 14 × 13 = 182 pairings of double sub-units are investigated, and d) six of these actually exhibit geometrical similarity.*

In order to establish whether any of the m double sub-units of the first structure is similar to any of the m double sub-units of the second structure, we generate all $m \times n$ possible pairs, as illustrated in Fig. 2c.

Each pair are tested by computing all absolute difference between corresponding entries x_i and $\underline{x}_i$ in their parameter lists. A schematic representation of this procedure is shown in Fig. 3. The sign of a torsion angle changes when the fragment is inverted. Thus, in order to account for the accidental choice of the molecule in structures with centres of inversion and/or mirror planes, we also calculate the absolute sum of corresponding entries in the *tor* lists.

These four lists of differences are then used to compute mean values δ_{ang}, δ_{dhd}, $\delta_{tor}/\delta_{tor*}$ which as *a whole* give an impression of the dissimilarity of the two double sub-units {K, S} and {$\underline{K}$, $\underline{S}$}. The general formula is

$$\delta = 1/M \sum_{i=1}^{M} |x_i - \underline{x}_i| \ ,$$

where x_i and $\underline{x}_i$ are the i-th entries of the parameter lists for {K, S} and {$\underline{K}$, $\underline{S}$}, respectively, and M is the total number of list entries. The δ values are compared against adjustable cut-off parameters Δ_{ang}, Δ_{dhd} and Δ_{tor}, and we consider two double sub-units to be similar if

$$(\delta_{ang} < \Delta_{ang}) \ \wedge \ (\delta_{dhd} < \Delta_{dhd}) \ \wedge \ [(\delta_{tor} < \Delta_{tor}) \vee (\delta_{tor*} < \Delta_{tor})] \Rightarrow \{K, S\} \sim \{\underline{K}, \underline{S}\}.$$

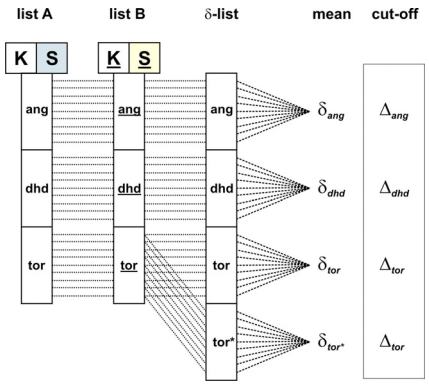


Fig. 3: *Comparison of the cluster sub-units {K, S} and {$\underline{K}$, $\underline{S}$}. a) List A with angles, dihedral and torsion angles representing {K, S}. b) The corresponding list B for {$\underline{K}$, $\underline{S}$}. c) The absolute difference for each pair of corresponding entries of lists A and B is computed. d) The mean values δ for each part of the list are computed and e) compared with cutoff parameters.*

The three panels in Fig. 4 show a typical distribution of δ values.[1b] Each data point corresponds to one of 196 sub-unit combinations (m, n = 14). The green points near 0° derived from six combinations that satisfy condition (2), all other combinations are represented by blue or red points.

In general, either of the following three cases will occur at this stage:

1. no similar double sub-units found ⇒ the two structures exhibit no packing similarity ⇒ the comparison is finished
2. a single similar double sub-unit found ⇒ this sub-unit is identical with the SC itself ⇒ the comparison is also finished
3. a set of similar double sub-units found ⇒ we need to establish whether they are actually parts of a larger seed

We continue in the next section with the treatment of case 3.

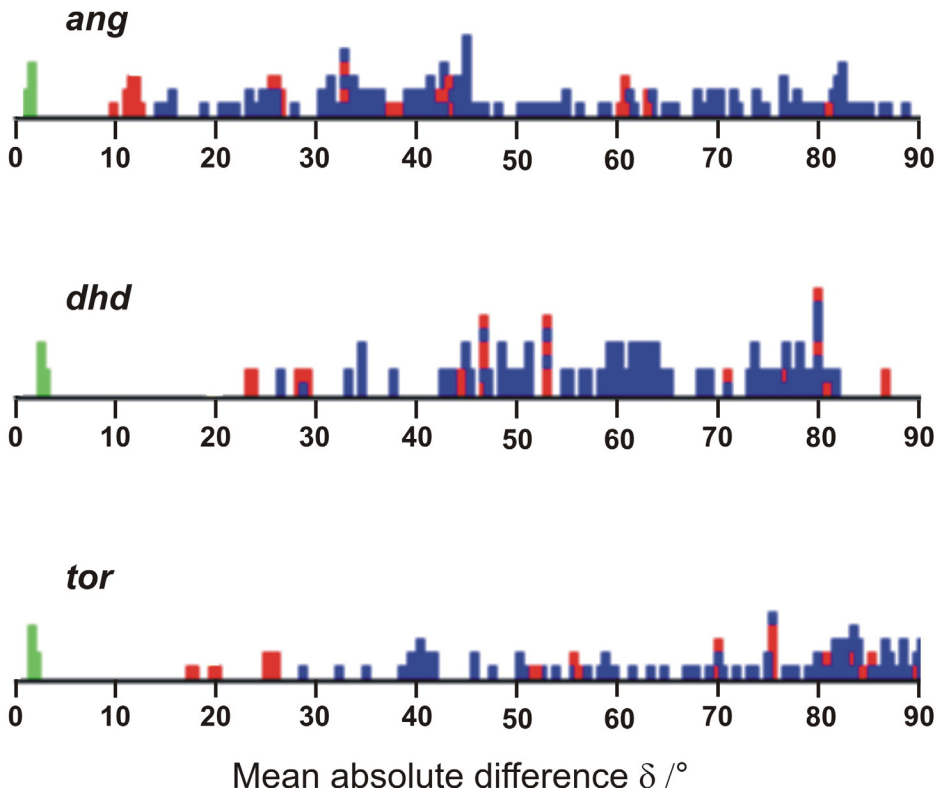


Fig. 4: *Distribution of δ values calculated for 196 combinations of double sub-units.*[1b] *The green data points near 0° belong to six pairs with a {K, S} ~ {K, S} relationship, data points for all other pairs are coloured blue and red. Note that a wide gaps separates the green points from the rest.*

Assembly of the supramolecular construct

We have now a certain number (at least two) of pairs of the type {K, S} ~ {K, S} and want to find out whether it is possible to join any of them so that pairs of larger cluster subunits with similar geometry are obtained. Merging two pairs of double sub-units gives a pair triple sub-units, {K, S_1, S_2} and {K, S_1, S_2}. From x pairs of double sub-units, we obtain $x^2 - x \ / 2$ triple pairs. This is illustrated in Fig, 5a. For each pair we will test whether

{K, S_1, S_2} ~ {K, S_1, S_2}

is true a statement (Fig. 5b). The test procedure is very similar to that outlined above for double sub-units. Again, we generate parameter lists. However, each parameter is now defined by points originating from three rather than two molecules. Again, we calculate the mean differences of corresponding list entries. A set of dissimilarity parameters δ_{ang}, δ_{dhd} and δ_{tor} / δ_{tor*} is computed and compared against the respective cut-off parameter Δ, and

$$(\delta_{ang} < \Delta_{ang}) \ \wedge \ (\delta_{dhd} < \Delta_{dhd}) \ \wedge \ [(\delta_{tor} < \Delta_{tor}) \vee (\delta_{tor*} < \Delta_{tor})] \Rightarrow \{K, S_1, S_2\} \sim \{K, S_1, S_2\}.$$

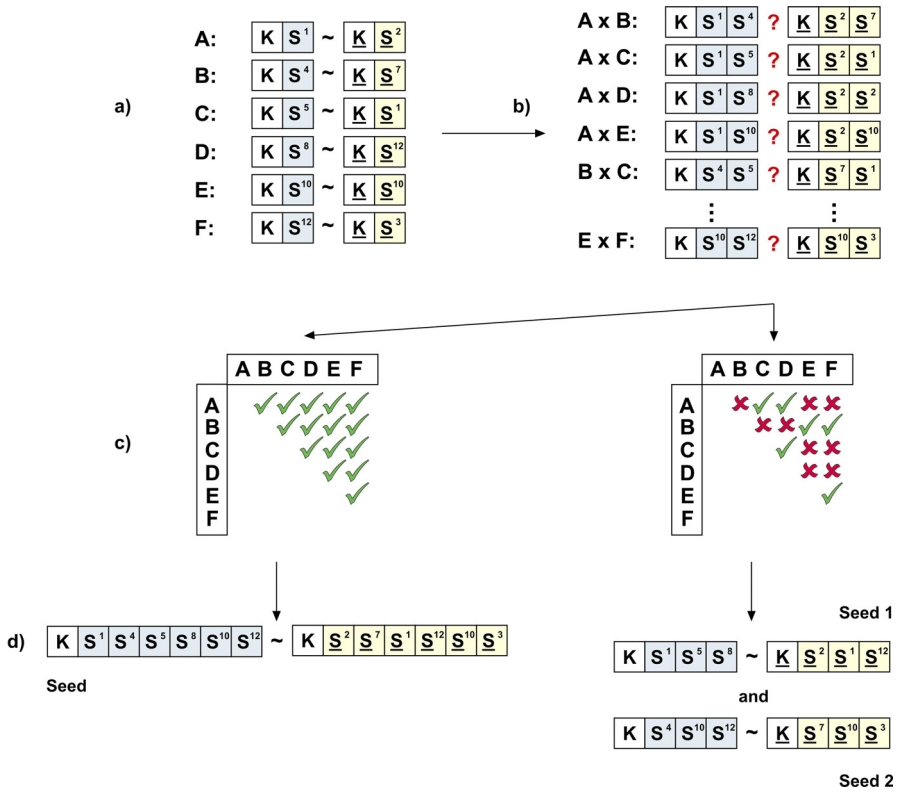


Fig. 5: *Continued from Fig. 2. a)* ***A*** *–* ***F****, the six {K, S} ~ {K, S} pairs from Fig. 2d. b) 15 triple sub-units obtained by pairing of* ***A*** *–* ***F****. c) and d) Two possible results of the subsequent comparison of these triple subunits. Left: All corresponding subunits are similar, i.e.* ***A*** *–* ***F*** *coexist in one seed. Right:* ***ACD*** *and* ***BEF*** *can only coexist in two distinct groups, leading to two seeds.*

We determine all pairs of similar triple clusters and use this information to assemble larger cluster sub-units, the seed of the SC. The left panel of Figs. 5c and 5d shows a case where all double subunits coexist in a single seed. By contrast, the right panel shows a case where certain combinations lead to dissimilar triple subunits. As a consequence, we obtain two distinct seeds, each containing three shell molecules.

This separation could well be genuine, but it could also be the result of cut-off parameters being set too low or to high. Then either too few or too many relationships are recognised as similar. Any such problems can be resolved by repeating the procedure with a different set of Δ-parameters. Finally, the seed can be expanded into the SC using its symmetry operations. At this point, information about the dimensionality and the corresponding base vectors of the SC can be obtained easily. The investigation of a series of N crystal structures consists of $N^2 - N / 2$ individual cycles of this kind, and the same COSP are used throughout the investigation.

Summary

The representation of crystal structures as clusters of molecules enables us to design powerful tools that permit a comparison even if the underlying structures crystallise in different space groups and with different Z'. This is useful for the analysis of polymorphic forms and predicted structures and provides a starting point for more complex investigations of large series of related structures.

References

1 T. Gelbrich, M. B. Hursthouse, *Cryst. Eng. Comm.* 2005, **7**, 324. a) Case study 4. b) Case study 1.
2 T. Gelbrich, M. B. Hursthouse, *Cryst. Eng. Comm.* 2006, **8**, 448.
3 R. M. Vrcelj, J. N. Sherwood, A. R. Kennedy, H. G. Gallagher, T. Gelbrich, *Cryst. Growth Des*. 2003, **3**, 1027.
4 S. Gerber, H. Krautscheid, T. Gelbrich, H. Vollmer, *Z. Anorg. Allg. Chem.* 2004, **630**, 1427.
5 T. Gelbrich, T. L. Threlfall, S. Huth, E. Seeger, M. B. Hursthouse, *Z. Anorg. Allg. Chem.* 2004, **630**, 1451.
6 B. Murphy, M. Aljabri, A. M. Ahmed, G. Murphy, B. J. Hathaway, M. E. Light, T. Gelbrich, M. B. Hursthouse, *Dalton Trans.* 2006, 357.

The Pixel module of the OPiX computer program package: affordable calculation of intermolecular interaction energies for large organic molecules and crystals

Angelo Gavezzotti
Dipartimento di Chimica Strutturale e Stereochimica Inorganica, Università di Milano, via Venezian 21, 20133 Milano (Italy), E-mail: angelo.gavezzotti@unimi.it , WWW: http://users.unimi.it/gavezzot/

Introduction

The Pixel method is a recently introduced procedure for the calculation of intermolecular interaction energies. The method is based on a semi-empirical computational procedure in which all the charge density of a molecule is taken into account rather than just a few nuclei and point charges. Electrostatic and repulsion terms are included in the calculations, but dispersion and polarization terms are also included because the methodology is mainly intended for the study of organic molecular assemblies rather than minerals or ionic solids.

Theoretical chemistry cannot continue to be based on calculations on just formic acid and urea, and a second characteristic of the Pixel method is its ready applicability to molecular systems of genuine chemical interest, say, a 50-atom molecule. The method is computationally demanding: in applying it to naphthalene, for example, one switches from an 18-site description of the molecule in the atom-atom potential scheme to a 20,000-site description. This is only possible now that computer-speeds have reached the level of teraflops per second

Pixel calculations are carried out with a submodule of the OPiX package, which can also perform a number of other tasks, including packing analysis and packing structure prediction. Use of the standard files in OPiX format greatly facilitates also the use of the Pixel module. A scheme and a brief description of the OPiX package follow.

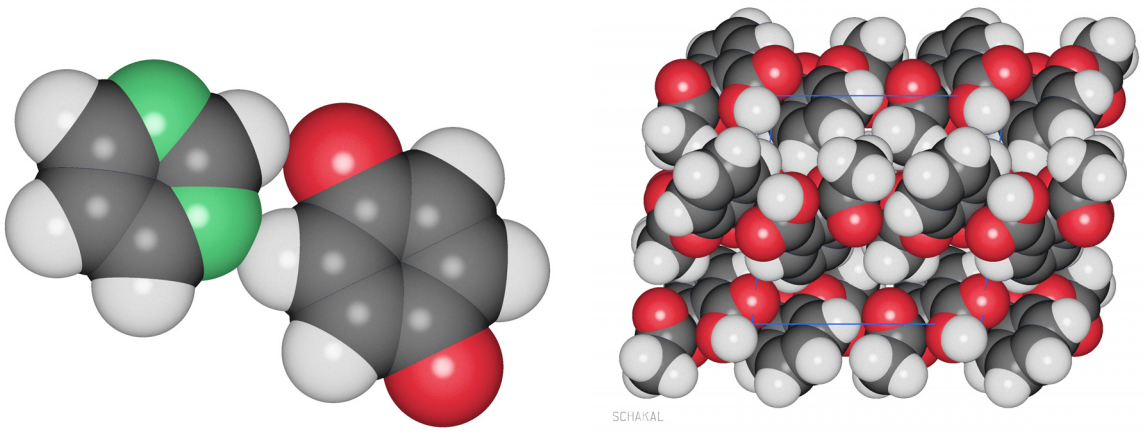


Fig 1: *a) calculate the full interaction profile for this dimer in 3 minutes... or b) ..or the lattice energy of aspirin in 10 minutes!*

Pixel calculations are carried out with a submodule of the OPiX package, which can also perform a number of other tasks, including packing analysis and packing structure prediction. Use of the standard files in OPiX format greatly facilitates also the use of the Pixel module. A scheme and a brief description of the OPiX package follow.

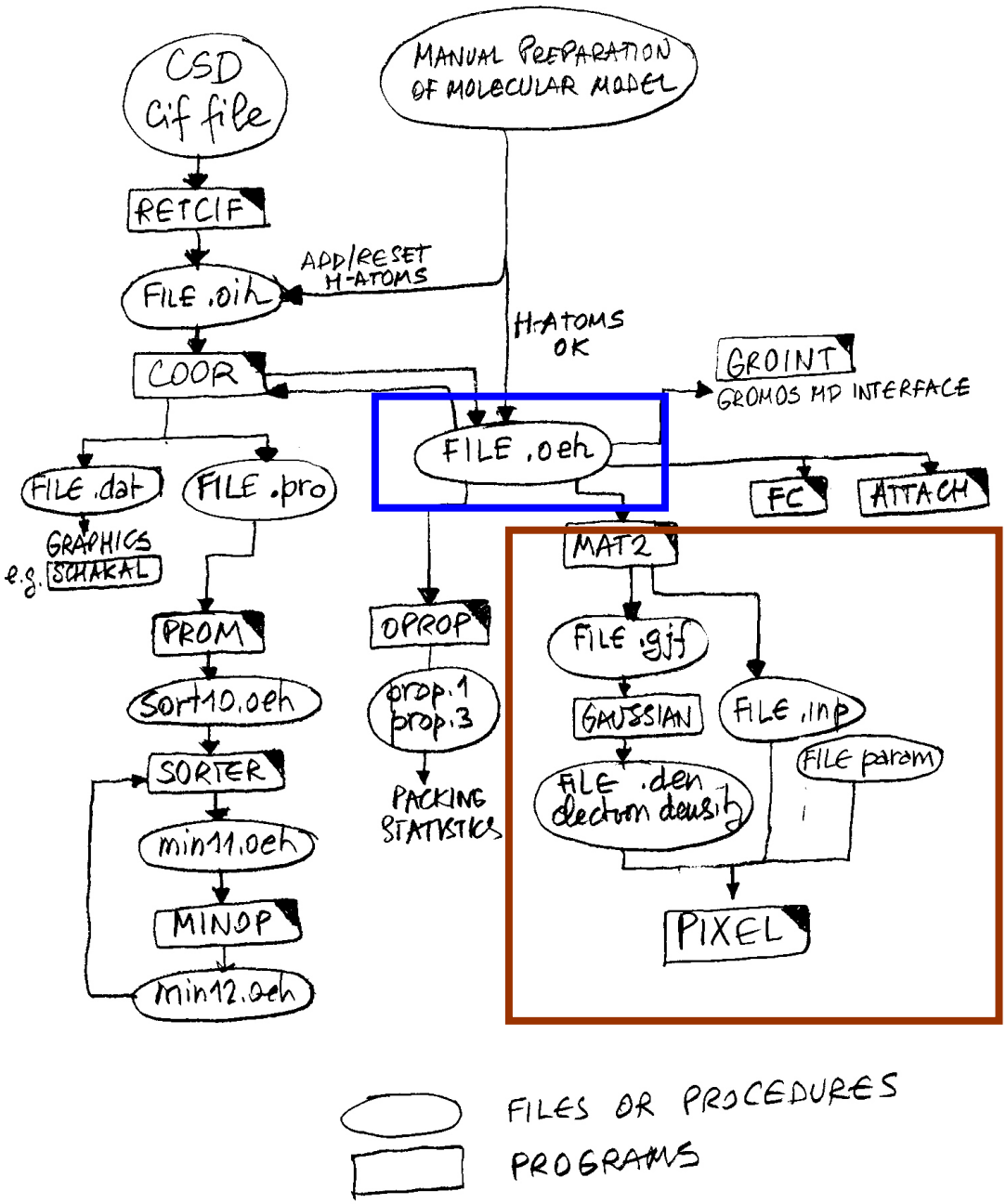


Fig 2: *OVERVIEW OF THE OPiX SYSTEM, blue: main input file ; red: the Pixel modules; View more detail at, and download files from: http://users.unimi.it/gavezzot/*

2. OPiX: General description

OPiX is a program package for the calculation of molecular and intermolecular properties. It is written in featureless Fortran and it has been compiled and used on all types of computers. OPiX uses file-to file input options, and there are no windows or menus.

3. Format of main input structural file

Molecular and crystal information can be given in one of the following file formats:

format 1 (filename extension **.oih**): atomic coordinates and crystal data (if any), implicit hydrogen atom positions. The positions of non-hydrogen nuclei are given as coordinates, while the positions of hydrogen

nuclei are given as a series of commands for the calculation of explicit coordinates for each hydrogen, using standard geometrical criteria from the position of non-hydrogen nuclei. Given the importance of H-atom positions in crystal energy calculations, this renormalization of hydrogen atom positions is essential when X-ray crystal atomic coordinates are used. Pixel has been calibrated using renormalized C-H distances of 1.08 Å or O-H and N-H distances of 1.0 Å.

format 2 (filename extension **.oeh**): atomic coordinates and crystal data (if any), explicit hydrogen atom positions. The positions of all nuclei are given as explicit coordinates. The COOR module converts .oih into .oeh files, i.e. calculates H-atom coordinates from the codes in the .oih file.

A .oeh file may contain atomic coordinates of one or more gas-phase molecule(s), or, if it is being used for a calculation on a crystal structure, the atomic coordinates of one reference molecular unit, plus cell dimensions and space group information. In the latter case, a cluster of molecules representing the crystal may be generated (this will not be possible when the structure is polymeric and the cell contents are covalently bound to surrounding cells). A .oih or .oeh file must contain coordinates for a full molecule, so for those crystals in which the asymmetric unit is a fraction of the molecule, the entire molecule must be reconstructed and the appropriate space subgroup must be used.

More detail, and a description of the PROM module (polymorph generator/predictor) and OPROP module (crystal packing properties and statistics) can be found in the documentation on the author's website (docs subdirectory of the opix directory): http://users.unimi.it/gavezzot/ .

4. The PIXEL module: Calculation of intermolecular energies by the semi-classical density sums (SCDS-pixel) method

The **Pixel modules** (PIXELC or PIXELD, see below) calculate the Coulombic, polarization, dispersion and repulsion energies between separate, rigid molecules. The interacting molecules in these calculations are represented by charge density distributions, usually calculated using the program GAUSSIAN (employing the CUBE option). Each elementary density volume with its charge is called a *pixel*. The original reference coordinate frame for position of the atomic nuclei and of the electron density pixels is the same one used in the GAUSSIAN calculation. No intramolecular energies are calculated. An additional module, MAT2 is used for the preparation of input files.

The electron density output by GAUSSIAN is encoded in an array containing a million or more pixels, and in order to reduce the run-time of subsequent calculations these are 'condensed' into larger pixels, yielding an array containing around 10-to-20000 values. The condensation level can be controlled by the user. A model of the set of interacting molecules also needs to be constructed: this might be a simple dimer or a larger array of molecules generated using crystallographic information. Within the set of interacting molecules there will be contact and overlap between undeformed densities, and interaction energies are obtained pixel-by-pixel summations of Coulombic, linear polarization, and dispersion (London-type) terms; repulsion is proportional to the overall integral between charge densities. The total energy is the sum of these four terms. The equations used to calculate the individual energy terms are given in refs. 5 and 13 in the Bibliography. Four parameters are needed in the calculation of the dispersion, polarization and repulsion energies; optimized values for these are given in the references; they have been hard-coded into the program as defaults, though these can be over-ridden by the user.

The Pixel method shares with calculations based on atom-atom potential calculations the advantage that explicit values are obtained for Coulombic, polarization, dispersion and repulsion terms. While it is not possible to calculate intramolecular (conformational) energies, treat dynamics or change or polarize the charge distribution, the PIXEL method has a number of important advantages over more conventional procedures. A molecule is treated as a 10,000-site object rather than a N(atom)-site object, and the method might be regarded as equivalent to an infinitely large multipolar expansion. In calculations using

atom-atom potentials each atom must be assigned a charge, whereas in the PIXEL method the Coulombic energy is parameter-less and so the Pixel Coulombic energy can be regarded as exact as the wavefunction used to calculate it. Penetration energies can also be calculated.

Two modules make up the facilities for carrying out PIXEL calculation. These are named PIXELD and PIXELC, and which is used depends on the type of problem being analysed:

PIXELD is used for clusters made of any number of molecules of up to two molecular species, the first species is referred to as *the solute*, the second species as *the solvent*. The position of each molecule in the cluster is obtained by transforming the original coordinates for nuclei and pixels by an orientation matrix constructed from three Euler angles, and by a displacement vector. These are expressed in the in the same coordinate frame as used in the GAUSSIAN calculation, and must be derived by the user.

PIXELC is used for calculations on a crystal made of only one molecular species, one molecule per asymmetric unit. The matrix operation that transforms from the molecular reference frame to coordinates in the crystal structure is obtained by use of a separate module MAT2; the positions of molecules in the cluster that represents the crystal structure are generated automatically using cell parameters, space group matrices, and a cutoff threshold for the distance between centers of mass of the central and surrounding molecules. Note that it is not possible to treat crystals containing more than one molecule per asymmetric unit.

In both cases, it is convenient to start with a .oeh data file which contains the molecular (and, if needed, the crystal) information. The minimization of the cluster energy or of the lattice energy can be carried out with a steepest-descent or a Symplex algorithm.

4.1 Program inputs and outputs

The input to Pixel consists of:

1) an electron density file for the solute (and one for the solvent molecule if any), extension .den
2) a file with the molecular specifications, extension .inp.
3) a file with the input parameters of the calculation, pixel.pmt

Besides, a .gjf file is needed for input to GAUSSIAN. Both .inp and .gjf file are prepared by submodule MAT2 from a .oeh file.

The program outputs are:

1) a listing-file with the results of the calculation; extension .oxp;
2) for the cluster calculation with PIXELD, a file with atomic coordinates; extension .dat
3) when a lattice energy optimization is carried out, a file (.oeh format) with the final crystal data; extension .pxm.

4.2 How to run a pixel calculation

1) Prepare a molecular model for the solute molecule (*x,y,z* coordinates) and set up a .oeh file. Cif files in the format output by the Cambridge Database can be converted into .oeh files using the module RETCIF-COOR, see scheme 1;

2) Run the MAT2 module. MAT2 prepares the Pixel input file .inp and the GAUSSIAN input file .gjf. MAT2 also calculates the transformation between original coordinates and a coordinate frame whose origin is the molecular center of charges with axes directed along the principal charge moments. MAT2 will also calculate automatically the limits of the electron density box and, for the crystal case, also the

matrix transformation that correlates the molecular coordinate frame and the crystal coordinate frame. MAT2 will also recognize atoms and assign atomic polarizabilities.

3) Run GAUSSIAN for the electron density calculation. The standard use requires a valence only density (option 'frozencore' in GAUSSIAN). If a MO package different from GAUSSIAN is used, the input data and the output electron density file must be converted accordingly. The parameters used in Pixel have been optimized using densities from MP2 6-31G** calculations.

4) Repeat steps 1 to 3 for the solvent molecule (the second molecular species), if there is one.

5) Prepare the parameter file and then run Pixel. The parameter file contains the parameters of the theory and a few indicators for condensing the molecular charge density

4.3 Options

4.3.1 Case a), a cluster of molecules, PIXELD module

4.3.1.1 Option i) calculate the energy for each set of positional parameters, three center of mass coordinates and three orientation angles for each molecule, specified from input one after the other. The values for the translational and the rotational displacements must be found by geometrical considerations; refer to the atomic coordinates in the GAUSSIAN output file. The first molecule is usually the undisplaced reference molecule, others are translated and rotated as needed.

4.3.1.2 Option ii) calculate the energy for a simultaneous variation of up to 5 parameters in a given range;

4.3.1.3 Option iii) Optimize the cluster energy by some cycles of steepest descent and some cycles of symplex. As expected, the optimization procedure is efficient very far from the minimum, but becomes critical if an energy valley is very shallow. Use automatic minimization far from the minimum, followed by a scan of the energy surface using option ii in the neighbourhood of shallow minima.

4.3.1.4 Option iv) PIXELD can be used also to calculate the energy of dimers or oligomers extracted from a crystal structure. In this case the input is exactly as for the crystal case (see below), but only the symmetry operations corresponding to the molecules in the oligomer must be given.

4.3.2 Case b), a crystal, PIXELC module

Molecules whose distance from the central one is in a prescribed range are automatically included in crystal model. Typically, one has 100-150 molecules in the cluster for which the Coulombic, polarization, dispersion and repulsion energies are calculated.

4.4 A brief description of the Pixel output files

While the program is running, some output appears on the screen (formal unit 6). The file output is directed to file pixel.oxp. This has a title, some echo of input parameters, and detail on the screenout and condensation procedures carried out on the original electron density, as well as of the subdivision of pixels among atomic basins. Then the output has the polarization energy at each molecule, and finally the Pixel energies: ec, electrostatic; ep, polarization; ed, dispersion; er, repulsion; et, total. The next line has the 6-exp force field (FF) energies: repulsion, attraction, total, point-charge energy, total+point charge energy. When this printout carries the 'lattice' tag, the Coulomb, dispersion, and repulsion energies are divided by two so the total is the computational equivalent of the enthalpy of sublimation.

5. What you can do with Pixel: Worked examples and output explanations

5.1 Calculation of electron densities and parameter file

A sample input file for GAUSSIAN for acetylene is:

```
#MP2/6-31G** guess=core nosym density=MP2 pop=esp cube=cards cube=frozencore
 acetylene
 0 1
 C     0.000000     0.000000    -0.600000
 C     0.000000     0.000000     0.600000
 H     0.000000     0.000000    -1.660000
 H     0.000000     0.000000     1.660000
c:\opix\dens\acetylen.den
     0 -3.800000    -3.800000    -4.280000
    96  0.080000     0.000000     0.000000
    96  0.000000     0.080000     0.000000
   108  0.000000     0.000000     0.080000
```

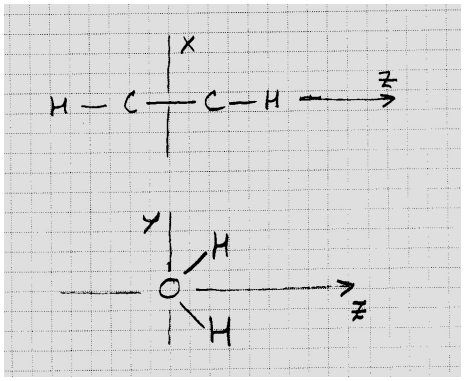


Fig 3: *Axis orientation of acetylene and water*

The Pixel parameter file: if the default version of the theory is desired, all these inputs are zero

```
0.00 3.00 150.0 4800.0 1200.0 0      the four parameters of the theory
   3 0 0.000001 99.0                 electron density trimmers, solute
   3 0 0.000001 99.0                 electron density trimmers, solvent
```

5.2 Example 1: Parallel acetylene dimer

The first example concerns an acetylene dimer with two molecules stacked along axis *x*, at distances of 3.35 and 4.0 Å (see 4.3.1.1). The input file is:

```
  acetylene dimer parallel          title
   2   0   4   0                    guide integers
   0.000   0.000
   1   3 -0.2000  0.0000            atomic information: atom number, atom type
   2   3 -0.2000  0.0000            (3=C, 1=H), charge and polarisability. The
   3   1  0.2000  0.0000            zeros given for polarizabiltiy that
   4   1  0.2000  0.0000            defaults should be used.
   0.000  0.000  0.000 0.000   0.000   0.000   1. position of 1st molecule
   3.350  0.000  0.000 0.000   0.000   0.000   1. position of 2nd molecule
    0  0
   0.000  0.000  0.000 0.000   0.000   0.000   1.
   4.000  0.000  0.000 0.000   0.000   0.000   1.
   99 99
```

The energy is calculated for only two distances and the output file is:

```
Welcome to Pixel DIMER version
  acetylene dimer parallel
Variable Krep,parameters Edisp,Epol,Erep
                   3.000   150.00   4800.00  1200.000
 Using damp.atomic ionization potent. for Edisp
  First molecule (solute) data
ionization potential and charge    0.0000    0.00
Atoms
  type, Z, x,y,z, point charge, polarizability, vdW R
    3    4.00     0.00000   0.00000  -0.59999 -0.2000   1.35    1.77
    3    4.00     0.00000   0.00000   0.59999 -0.2000   1.35    1.77
    1    1.00     0.00000   0.00000  -1.65998  0.2000   0.39    1.10
    1    1.00     0.00000   0.00000   1.65998  0.2000   0.39    1.10

Density file title  acetylene
 min and max original density  0.0000E+00  0.8025E+01      atomic basins
atom   1 basin and real charge     4.207     4.000
atom   2 basin and real charge     4.207     4.000
atom   3 basin and real charge     0.792     1.000
atom   4 basin and real charge     0.792     1.000
  Condensation level    3
  Density steps,original   96   96  108 and condensed   32   32   36
steps and pixel vol(A)     0.2400   0.2400   0.2400   0.01382
original electron number   9.99817  remaining-pixels     9.99652   9940
q min and max  0.1000E-05     99.00
screening: electrons out low and high  0.1652E-02  0.0000E+00
atom   1 condens.basin and real charge    -4.236     4.000
atom   2 condens.basin and real charge    -4.236     4.000
atom   3 condens.basin and real charge    -0.764     1.000
atom   4 condens.basin and real charge    -0.764     1.000
polarizability,raw, tot, renorm  0.35546E+01  0.34800E+01  0.34800E+01
renormalized total charges     10.000000  -10.000000
no. of charge points per atom
   1 3882  2 3882  3 1088  4 1088

===== Start energy calculations =====
 collision parameter     0.120
Molecular cluster, positions, c.o.m. and euler angles
 solute     1   0.0000   0.0000   0.0000    0.00    0.00    0.00   1.0
 solute     2   3.3500   0.0000   0.0000    0.00    0.00    0.00   1.0
A...A  A...B  B...B total energies
coul       3.1     0.0     0.0     3.1
disp      -5.3     0.0     0.0    -5.3
rep        9.2     0.0     0.0     9.2
total polarization    -1.4 total energy     5.6         total Pixel energy
 FF cluster e6r eqq etot    -1.0     2.9   1.8  atom-atom force field energy

Molecular cluster, positions, c.o.m. and euler angles
 solute     1   0.0000   0.0000   0.0000    0.00    0.00    0.00   1.0
 solute     2   4.0000   0.0000   0.0000    0.00    0.00    0.00   1.0
A...A  A...B  B...B total energies
coul       2.5     0.0     0.0     2.5        the first three columns are
disp      -1.8     0.0     0.0    -1.8    solute-solute, solvent-solute and
rep        0.6     0.0     0.0     0.6      solvent -solvent energies
total polarization    -0.2 total energy     1.1
 FF cluster e6r eqq etot    -2.1     1.5    -0.6
```

5.3 Example 2: acetylene-$(H_2O)_2$ trimer

The hydrogen atoms of the acetylene molecule point at the oxygen atom of two water molecules along the bisector of the HOH angle:

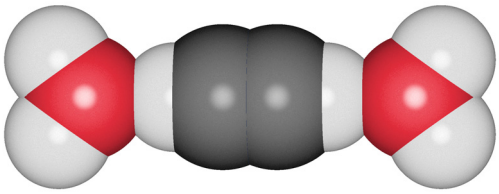


Fig 4: *acetylene-$(H_2O)_2$ trimer ; z-axis→*

The first water molecule is just displaced along +*z*, while the second one is inverted and displaced along -*z*. The H...O distances are varied by having the z-coordinates of the water molecules vary from 3.66 to 4.06 and from -4.06 to -3.66 in steps of 0.2 Å . See in 4.3.1.2. The input file is:

```
acetylene-water(2) trimer
  1   2   4   3
  0.000   0.000
  1   3 -0.2000  0.0000                  acetylene atomic info
  2   3 -0.2000  0.0000
  3   1  0.2000  0.0000
  4   1  0.2000  0.0000
  0.000   0.000
  1  16 -0.8000  0.0000                  water atomic info
  2  25  0.4000  0.0000
  3  25  0.4000  0.0000
0.000   0.000   0.000    0.00   0.00   0.00   1.0     starting positions
0.000   0.000   3.6600   0.00   0.00   0.00   1.0
0.000   0.000  -4.0600   0.00   0.00   0.00  -1.0
  -1  0
0.  0.  0.     0.  0.  0.                  final positions
0.  0.  4.06   0.  0.  0.
0.  0. -3.66   0.  0.  0.
0.  0.  0.     0.  0.  0.                   steps
0.  0.  0.2    0.  0.  0.
0.  0.  0.2    0.  0.  0.
```

The last part of the output file has the interaction energies

```
   1 variable parameters   3.660  -4.060
  cluster ec ep ed er et   -37.1   -10.2    -9.2    30.9   -25.7
 FF cluster e6r eqq etot    12.2   -20.7    -8.5
   2 variable parameters   3.660  -3.860
  cluster ec ep ed er et   -42.2   -12.1   -10.5    38.1   -26.7
 FF cluster e6r eqq etot    15.5   -22.7    -7.2
   3 variable parameters   3.660  -3.660
  cluster ec ep ed er et   -51.1   -15.9   -12.2    52.1   -27.1
 FF cluster e6r eqq etot    25.8   -25.5     0.2
   4 variable parameters   3.860  -4.060
  cluster ec ep ed er et   -28.2    -6.3    -7.6    16.9   -25.2
 FF cluster e6r eqq etot     2.0   -17.9   -15.9
   5 variable parameters   3.860  -3.860
  cluster ec ep ed er et   -33.3    -8.3    -8.8    24.2   -26.2
 FF cluster e6r eqq etot     5.3   -19.9   -14.6
   6 variable parameters   3.860  -3.660
  cluster ec ep ed er et   -42.2   -12.1   -10.5    38.1   -26.7
 FF cluster e6r eqq etot    15.5   -22.7    -7.2
   7 variable parameters   4.060  -4.060
  cluster ec ep ed er et   -23.1    -4.4    -6.3     9.6   -24.2
 FF cluster e6r eqq etot    -1.3   -15.8   -17.1
```

5.4 Example 3: optimization of the acetylene-water π-dimer

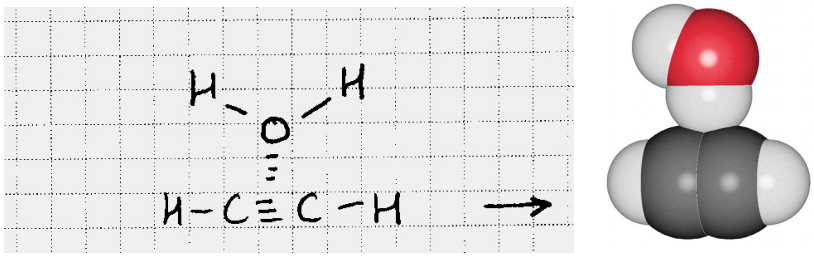


Fig 4: *acetylene-water π-dimer*

See in 4.3.1.3. The input file is:

```
 acetylene-water O..pi dimer
  1   1   4   3
  0.000   0.000
  1   3 -0.2000  0.0000
  2   3 -0.2000  0.0000
  3   1  0.2000  0.0000
  4   1  0.2000  0.0000
  0.000   0.000
  1  16 -0.8000  0.0000
  2  25  0.4000  0.0000
  3  25  0.4000  0.0000
0.000   0.000   0.000    0.00   0.00   0.00   1.0
0.000   2.600   0.000   90.00   0.00   0.00   1.0
   2 10                        two cycles st.descent, 10 cycles Symplex
0 0 0 0 0 0  0 1 0 1 0 0       optimization tags: acetylene molecule fixed,
0.1 1. 2. 20.0 0.001           water molecule: vary only y-coordinate
                               and in-plane rotation angle
```

The starting configuration is is obtained by rotating the water molecule 90° around *x* and displacing by 2.6 Å along *y*, as shown on the left. These two parameters are then optimized and the final configuration is seen on the right. Steepest descent works well far from the minimum, Symplex is more performing close to the minimum. Pixel reasonably predicts that the π-system of acetylene will be approached by the H-atom of the water molecule. The last part of the output shows the progress of the Symplex optimization:

```
current parameters
       0.000     0.000     0.000     0.000     0.000     0.000
       0.000     3.435     0.000   100.095     0.000     0.000
  cluster ec ep ed er et    -4.0    -2.1    -4.3     5.0    -5.4
 current parameters
       0.000     0.000     0.000     0.000     0.000     0.000
       0.000     3.425     0.000   102.157     0.000     0.000
  cluster ec ep ed er et    -4.3    -2.3    -4.4     5.3    -5.7
 current parameters
       0.000     0.000     0.000     0.000     0.000     0.000
       0.000     3.356     0.000   101.282     0.000     0.000
  cluster ec ep ed er et    -4.7    -2.8    -4.9     7.2    -5.3
 current parameters
       0.000     0.000     0.000     0.000     0.000     0.000
       0.000     3.372     0.000   105.845     0.000     0.000
  cluster ec ep ed er et    -5.3    -3.0    -4.8     7.4    -5.7
 current parameters
       0.000     0.000     0.000     0.000     0.000     0.000
       0.000     3.353     0.000   109.970     0.000     0.000
  cluster ec ep ed er et    -6.3    -3.7    -5.1     8.7    -6.3
  max iteration number exceeded     10
 writing coord file for last parameters
      1     0.000     0.000     0.000       0.0       0.0       0.0
      1     0.000     3.353     0.000     110.0       0.0       0.0
```

```
 energy for last parameters
 A...A  A...B  B...B total energies
 coul       0.0    -6.3     0.0    -6.3  note:only solute-solvent energies are nonzero
 disp       0.0    -5.1     0.0    -5.1
 rep        0.0     8.7     0.0     8.7
 total polarization    -3.7 total energy    -6.3
  FF cluster e6r eqq etot     0.5    -4.1    -3.6
```

5.5 Example 4: Crystal lattice energy calculations

See in 4.3.2. The example is for the formic acid crystal. Files formac.oeh and formac.gjf are:

```
#FORMAC01****
   1    .000  0
 10.2410   3.5440   5.3560   90.00   90.00   90.00    crystal cell
     0
     5
   1    .1584    .3109    .1655  1  3    .0000       fractional
   2    .0834    .1423    .0023  1 18    .0000       atomic coords
   3    .2766    .3424    .1430  1 17    .0000
   4    .1410    .0393   -.1322  1 26    .0000
   5    .1131    .4292    .3300  1  1    .0000
     0
     4   4                      space group symmetry information
  1.00    .00    .00    .00   1.00    .00    .00    .00   1.00
   .00000    .00000    .00000
 -1.00    .00    .00    .00   1.00    .00    .00    .00   1.00
   .50000    .50000    .50000
  1.00    .00    .00    .00  -1.00    .00    .00    .00   1.00
   .50000    .50000    .00000
 -1.00    .00    .00    .00  -1.00    .00    .00    .00   1.00
   .00000    .00000    .50000
   3    0    0    0
   0
```

```
#MP2/6-31G** guess=core nosym density=MP2 pop=esp cube=cards cube=frozencore
FORMAC01****
 0 1
 C   -0.002289    0.409174    0.108307     atomic coords. in molecular
 O    0.005214   -0.084918   -1.102822     reference frame
 O    0.002073   -0.272312    1.122122
 H   -0.031114   -1.081109   -1.022557
 H   -0.013451    1.483901    0.218310

c:\opix\dens\FORMAC01.den
    0   -3.960000   -3.800000   -3.960000      electron density box
  100    0.080000    0.000000    0.000000
  100    0.000000    0.080000    0.000000
  100    0.000000    0.000000    0.080000
```

In this calculation, a rather small box radius (7.0 Å) is considered for a quick example, and the crystal cluster contains only 31 near-neighbor molecules. Usually, the radius may be of the order of 15-18 Å, and the crystal cluster may contain 100-150 molecules. The calculation with condensation level 5 lasts about one minute. The formac.inp Pixel input file is:

```
 FORMAC01****
   0  -1   5   0
   0.000   0.000
   1   3  0.0000  1.0500   atomic species, charge and polarizability
   2  18  0.0000  0.0000
   3  17  0.0000  0.0000
   4  26  0.0000  0.0000
   5   1  0.0000  0.0000
    0.0000   7.0000   35.0000  crystal box radius
   10.2410    3.5440    5.3560   90.0000   90.0000   90.0000   cell
  1.000  0.000  0.000            orientation matrices
```

```
 0.000  1.000  0.000
 0.000  0.000  1.000
 0.000  0.000  0.000
  -0.124704 -0.519479  0.845335   relates molecular to crystal
   0.863259  0.363191  0.350538   coordinates
  -0.489115  0.773456  0.403154
   1.742890   0.917232   0.525156
  4       space group symmetry
  1.00    0.00    0.00    0.00    1.00    0.00    0.00    0.00    1.00
 0.000000  0.000000  0.000000
 -1.00    0.00    0.00    0.00    1.00    0.00    0.00    0.00    1.00
 0.500000  0.500000  0.500000
  1.00    0.00    0.00    0.00   -1.00    0.00    0.00    0.00    1.00
 0.500000  0.500000  0.000000
 -1.00    0.00    0.00    0.00   -1.00    0.00    0.00    0.00    1.00
 0.000000  0.000000  0.500000
 99
```

The output file is:

```
 FORMAC01****
Variable Krep,parameters Edisp,Epol,Erep
                   3.000   150.00   4800.00  1200.000
 Using damp.atomic ionization potent. for Edisp
  molecular data
ionization potential and charge    0.0000    0.00
Atoms
  type, Z, x,y,z, point charge, polarizability, vdW R
    3    4.00    -0.00229   0.40917   0.10831  0.0000   1.05    1.77
   18    6.00     0.00521  -0.08492  -1.10281  0.0000   0.75    1.58
   17    6.00     0.00207  -0.27231   1.12211  0.0000   0.75    1.58
   26    1.00    -0.03111  -1.08109  -1.02254  0.0000   0.39    1.10
    1    1.00    -0.01345   1.48388   0.21831  0.0000   0.39    1.10
Density file title  FORMAC01****
 min and max original density  0.0000E+00  0.1833E+02
  Condensation level    5
  Density steps,original  100  100  100 and condensed   20   20   20
steps and pixel vol(A)     0.4000   0.4000   0.4000   0.06400
original electron number  17.99138  remaining-pixels    17.99105   2622
                                  (molecule is described by 2622 e-Pixels)
q min and max  0.1000E-05     99.00
screening: electrons out low and high  0.3369E-03  0.0000E+00
polarizability,raw, tot, renorm  0.34447E+01  0.33300E+01  0.33300E+01
renormalized total charges     18.000000  -18.000000

===== Start energy calculations =====
 collision parameter     0.200
 CRYSTAL; between   0.00   7.00  top cutoff  35.00
 cell parameters  10.241   3.544   5.356   90.00   90.00   90.00
Space group matrices
  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.000  0.000  0.000
 -1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.500  0.500  0.500
  1.0  0.0  0.0  0.0 -1.0  0.0  0.0  0.0  1.0    0.500  0.500  0.000
 -1.0  0.0  0.0  0.0 -1.0  0.0  0.0  0.0  1.0    0.000  0.000  0.500
euler angles and c.o.m.   62.5   29.3   98.2   1.7429   0.9172   0.5252
      31 molecules within crystal cutoff shell
overlap by atom types    1    3      0.000021
overlap by atom types    1   17      0.001296      atom-atom overlap integrals
overlap by atom types    1   18      0.000757      between charge densities
overlap by atom types    1   26      0.000006
overlap by atom types    3    1      0.000021      atom types: 1 hydrogen,
overlap by atom types    3    3      0.000319      3 carbon, 17 carbonyl O, 18 O,
overlap by atom types    3   17      0.001680      26 OH hydrogen
overlap by atom types    3   18      0.000927
overlap by atom types    3   26      0.000427
overlap by atom types   17    1      0.001296
overlap by atom types   17    3      0.001681
overlap by atom types   17   17      0.000234
overlap by atom types   17   18      0.000751
overlap by atom types   17   26      0.026061      (O)-H...O= overlap
overlap by atom types   18    1      0.000757
overlap by atom types   18    3      0.000927
```

```
 overlap by atom types    18   17      0.000753
 overlap by atom types    18   18      0.000283
 overlap by atom types    18   26      0.000023
 overlap by atom types    26    1      0.000006
 overlap by atom types    26    3      0.000427
 overlap by atom types    26   17      0.026050
 overlap by atom types    26   18      0.000023
   Epol,damp,no-damp  -43.91 -658.93
  FORMAC01****   lattice ec ep ed er et  -86.5  -43.9  -33.4  104.7   -59.1   Pixel energies
  FF lattice e6 er e6r eqq etot  -105.8    60.8   -45.0     0.0   -45.0 atom-atom energies
```

Note that energies are qualified as 'lattice': that means that the Coulombic, dispersion and repulsion energies have been divided by two to avoid double counting as usual in pairwise additive lattice summations. The polarization energy has not, since it is not counted twice. The auxiliary output file formac.mlc has detail of the molecule-molecule interactions, first the symmetry operations, and then a list of Coulombic, dispersion and repulsion molecule-molecule energies. This output can be used to spot the most important intermolecular interactions in the crystal structure.

5.6 Example 5: dimers from a crystal structure

See in 4.3.1.4. This example calculates the energies, including total polarization, for several dimers extracted from the formic acid crystal. There are only the two symmetry operations for each pair of molecules; the symmetry operation for the desired dimers can be obtained from a survey of file formac.mlc.

The last part of the input file is:

```
  2
  1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
 0.000000  0.000000  0.000000
 -1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
 0.500000  0.500000  0.500000
  2
  1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
 0.000000  0.000000  0.000000
  1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
 0.00000  1.00000  0.00000
     99
```

OUTPUT (LAST PART)

```
..................  as above for full crystal

  cell parameters  10.241   3.544   5.356   90.00   90.00   90.00
 euler angles and c.o.m.   62.5   29.3   98.2   1.7429   0.9172   0.5252
 Space group matrices
   1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.000  0.000  0.000
  -1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.500  0.500  0.500
       2 molecules within crystal cutoff shell
 A...A  A...B  B...B total energies
 coul     -77.0     0.0     0.0   -77.0
 disp     -13.3     0.0     0.0   -13.3
 rep       91.6     0.0     0.0    91.6
 total polarization   -37.4 total energy   -36.0
  FF cluster e6r eqq etot   -22.0     0.0   -22.0
 Space group matrices
   1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.000  0.000  0.000
   1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  1.0    0.000  1.000  0.000
       2 molecules within crystal cutoff shell
 A...A  A...B  B...B total energies
 coul       3.4     0.0     0.0     3.4
 disp      -5.4     0.0     0.0    -5.4
 rep        3.8     0.0     0.0     3.8
```

```
 total polarization    -1.0 total energy     0.9
  FF cluster e6r eqq etot    -5.3     0.0    -5.3
```

5.7 Example 6: Optimization of a crystal structure

Optimization of a crystal structure is done by optimizing the cell parameters and the rigid-body molecular position, i.e. three coordinates for the center of mass and three Euler angles for orientation. Steepest descent: very efficient far from minima, inefficient for fine optimization; Symplex: more efficient but requires a very high number of cycles, typically 300-500. This is a very time-consuming calculation.

```
INPUT (LAST PART):

..............as formac.inp above

4
   1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
  0.000000  0.000000  0.000000
  -1.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   1.00
  0.500000  0.500000  0.500000
   1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   1.00
  0.500000  0.500000  0.000000
  -1.00   0.00   0.00   0.00  -1.00   0.00   0.00   0.00   1.00
  0.000000  0.000000  0.500000
   2
  30   1  1  1   0  0  0   1  1  0   1  1  1    number of symplex cycles
                                                 optimization tags: optimize a,b,c
                                                 cell parameters, do not vary cell
                                                 angles (orthorhombic), do not optimize
                                                 z coordinate of center of mass
                                                 (space group polar along z)
```

Note the request of only 30 cycles of Simplex optimization (typical values are around 500). Note that positions along floating origin directions should not be optimised'.

6. Final comments

Running times are short: on a normal PC running under MS-DOS the calculation of the lattice energy of a 30-atom molecule may take some 30 minutes at condensation 4, and 10 minutes at condensation 5 (good enough for a reasonable estimate of the lattice energy). One point on the energy profile for a dimer of two 15-atom molecules is a matter of seconds. On a really fast or parallel machine, computing times become as comparable to the computing times for the atom-atom potential method were twenty years ago.

OPiX-Pixel is easy to use once you invest a minimum amount of time learning requirements for file formats etc. A number of examples are provided with the program package, and these should be carefully worked through. Do not be discouraged if the first attempts seem complicated. However, OPiX is by no means a black-box, push-button system, so *you* are in charge of the calculation, not the package.

The OPiX package is distributed free of charge to academic institutions and non-profit organizations. Redistribution of the software requires permission of the author. Use for Companies and other profit organizations is subject to conditions.

The program can be obtained from the web-site http://users.unimi.it/gavezzot/

The following literature citation should be used when publishing results obtained using the package:
A.Gavezzotti (2005) OPiX, *A computer program package for the calculation of intermolecular interactions and crystal energies*, University of Milano, together with the first two items of the following list.

Bibliography

The following papers contain a description of the method:
1. A. Gavezzotti, *J.Phys.Chem*. (2002) **B106**, 4145.
2. A. Gavezzotti, *J.Phys.Chem*. (2003) **B107**, 2344.
3. A. Gavezzotti, *CrystEngComm*, (2003) **5**, 429 and 2003, **5**, 439.
4. A.Gavezzotti, *Z. Krist.* (2005) **220**, 499.

The following papers describe applications of the method:
5. A. Gavezzotti, *J.Chem.Theor.Comput.* (2005) **1**,834.
6. J. D. Dunitz and A. Gavezzotti, *Angew. Chem. Int. Ed.* (2005) **44**, 2.
7. R.Boese, T.Clark and A.Gavezzotti, *Helv.Chim.Acta* (2003) **86**, 1085.
8. J.D.Dunitz, A.Gavezzotti, *Helv. Chim. Acta* (2003) **86**, 4073.
9. F. Demartin, G. Filippini, A. Gavezzotti, S. Rizzato. *Acta Cryst.*(2004) **B60**, 609.
10. L.Carlucci, A.Gavezzotti, *Chem. Eur. J.* (2005) **11**, 271.
11. A.Gavezzotti, *Structural Chemistry*, (2005) **16,** 177.
12. J.D.Dunitz, A.Gavezzotti, *Cryst.Growth Des.*(2005) **5**, 2180.
13. S. Bacchi, M. Benaglia, F. Cozzi, F. Demartin, G.Filippini, A. Gavezzotti, *Chem.Eur.J.* (2006) **12**, 3538;
14. J.Dunitz, B.Schweizer, *J.Chem.Theor.Comp.* (2006) **2**, 288. see also: Y.Ma, P.Politzer, *J.Chem.Phys.* (2004) **120**, 3152.

For extensive discussions of the Pixel method see also A.Gavezzotti, Molecular aggregation, Structure analysis and molecular simulation of crystals and liquids, Oxford University Press, Oxford 2006: http://www.oup.com/uk/catalogue/?ci=9780198570806.

Quantifying the Similarity of Crystal Structures

René de Gelder
Institute for Molecules and Materials, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, The Netherlands. - Email: R.deGelder@science.ru.nl ; WWW: http://www.crystallography.nl/

Introduction

Comparison of crystal structures and the quantification of structural similarity is far from trivial. A unique and practical representation of a crystal structure is not easy to define and sophisticated methods are needed to assess structural similarity, especially when structures are compared in a more global sense, *e.g.* in terms of packing principles. Comparing hundreds of thousands of structures, as present in the CSD database[1], needs clever automation, despite the speed of today's computers. On the other hand, with the increasing speed and data storage capacity of today's computers the time seems ripe for implementing automated procedures that look for hidden structural relations and principles that cannot be found by the usual search procedures in current database software.

The intention of this article is not to review all existing crystal structure comparison approaches and methods described in literature. Instead, a method for crystal structure comparison based on powder diffraction patterns is described, together with the source code to get the actual figure for the structural similarity. This method can easily be implemented as part of the usual software present in crystallography laboratories.

It is also discussed and demonstrated what automation of crystal structure comparison can do for the daily practice of a crystallographer and for crystal engineering research.

Why crystal structure comparison?

After solving a crystal structure the question is always whether the new structure is unique compared to the collection of structures already recorded in literature and databases. Although today's database software is very user-friendly one can easily miss closely related structures that might be of considerable interest to your research. This is not directly a shortcoming of the database software or the user but is a result of chemical interpretation and chemical variety that cannot be anticipated easily (see Fig. 1). Therefore, robust and objective search methods are still very welcome, also for answering the seemingly trivial question of structural uniqueness.

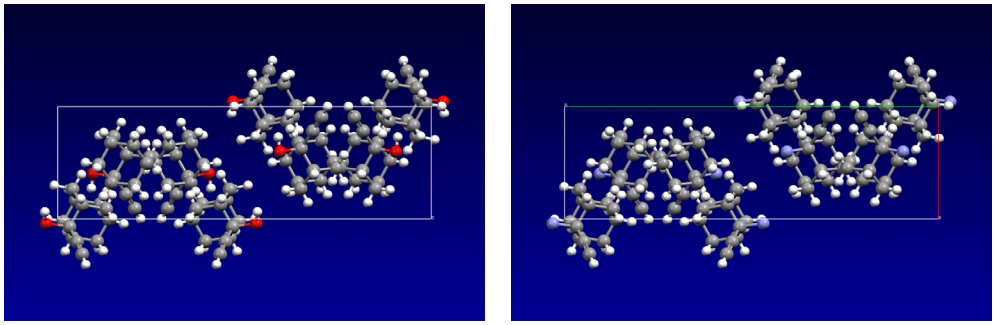


Fig 1: *Two isostructural compounds with a subtle difference in chemistry (CSD refcodes NAMKUH and BETXAZ)*

Comparing crystal structures is not only relevant to crystallographers who solved a new crystal structure. One can also be interested in the phenomena of polymorphism and/or isostructurality. Are there specific non-trivial families of structures in the CSD or in my in-house structural database? What atoms and groups can I replace in my molecule without disturbing the crystal packing? Replacing metals, as is well

known, often results in isomorphous compounds. Cl can often be replaced by F, I, Br or methyl. However, can replacement of other functional groups still lead to isostructurality? At least we would like to know about such cases (see Fig. 2).

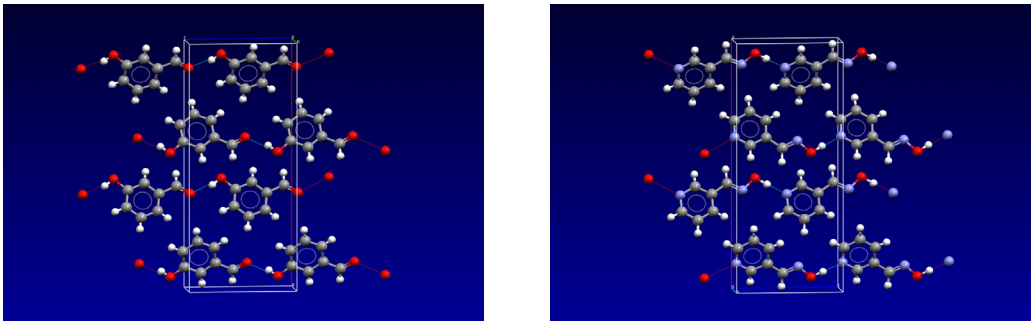


Fig 2: *Different molecules, the same crystal packing (CSD refcodes WOMXOL and XAYCIJ)*

In crystal engineering a correct classification of a set of structures can be essential for the understanding of the relation between physical properties and the underlying structure of materials. In the clustering stage of *ab initio* crystal structure prediction representative subsets have to be generated for which analysis and geometry optimization is feasible. The conclusion is that there are a lot of situations in which methods are needed that objectively quantify the similarity of crystal structures in a fast and reliable way.

The problem: unique descriptors of crystal structures

The usual procedure for finding related crystal structures in literature is defining a suitable query in crystallographic databases like the CSD, ICSD or PDB. Such a query can be based on the chemical composition, the unit cell, the space group, a chemical fragment, *etc*. Searches based on chemical composition are often too strict. Reduced unit-cell parameters can vary significantly with minor lattice distortions. On the other hand, searches on unit cells with a "safe" tolerance often give very long lists of hits. In Fig. 3a an example is given of a structure that initially was not recognized as a redetermination as a result of a difference in chemical interpretation (tetramer versus polymer) and a large anisotropic contraction of the unit cell going from room to low temperature.

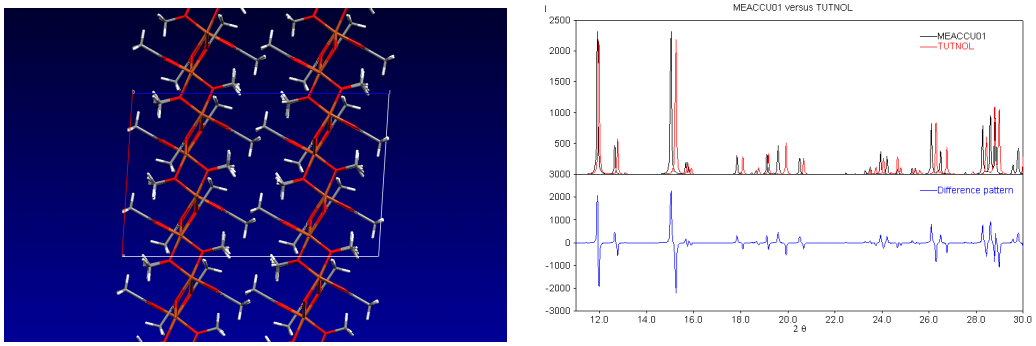


Fig 3: a) *structure initially (1996) interpreted as a tetramer (CSD refcode MEACCU01), later (2003) as a polymer (CSD refcode MEACCU02, initially TUTNOL)* **b)** *PXRD patterns for structure MEACCU01/02, measured at different temperatures*

Using the space group is risky when you have phase transitions or symmetry breaking as a result of a change in temperature or slight chemical modifications. Atoms, substituents or bonds in a chemical search fragment can be replaced by others to give compounds that show similar molecular structures and/or crystal packings. The problem, which is general for crystal structure comparison, is that you would like to have a descriptor for your structure that is not too strict in the chemical sense, is unique for the packing of the molecules and doesn't require a choice of origin or some other setting (which is the case for the combination of unit cell parameters and fractional coordinates).

The powder diffraction pattern as a global descriptor

There are global descriptors for crystal structures, like the radial distribution function or modifications of this function[2], that fulfill the requirements as outlined above. A powder diffraction pattern is also a global descriptor of a crystal structure, more precisely of the periodic electron density in a crystal. Such a descriptor can easily be calculated with many public domain programs (like Mercury[3]) and is therefore available to the crystallographer. Although invariable for the choice of origin or setting, in the case of powder diffraction patterns the positions of the peaks are very sensitive to small deviations in unit cell parameters. This means that strongly related structures may give (calculated) powder diffraction patterns that look similar from an overall point of view but differ significantly on a more local scale (see Fig. 3b). The same situation may occur for isomorphous compounds that differ only slightly in unit cell volume or unit cell shape. These compounds give calculated powder patterns that by visual inspection are definitely related and recognizable as isostructural compounds. In all those cases the calculation of a reliable and objective measure for the similarity or dissimilarity of the powder patterns is the essential and crucial step. With a pattern matching approach based on correlation functions[4], crystal structures can reliably be compared and classified on the basis of their PXRD patterns. In this article we discuss the relation between the *WCC* (Weighted Cross Correlation) criterion, for which the Fortran code will be given, and the well known *RMS* and R_{wp} criteria.

The comparison of powder diffraction patterns

The *RMS* (Root Mean Square) criterion is a well known criterion that is based on the sum of the squared differences between n observed and n calculated data values, respectively $y_i(obs)$ and $y_i(calc)$:

$$RMS = (\sum_{i=1,\,n} (y_i(obs)-y_i(calc))^2)^{1/2} \qquad [I]$$

In principle, with the *RMS* criterion one measures the length of the difference vector $\boldsymbol{d}$, $|\boldsymbol{d}|$, spanned by the observed data vector $\boldsymbol{y}(obs)$ and calculated data vector $\boldsymbol{y}(calc)$.

The *RMS* criterion becomes the sine of the angle between the vectors $\boldsymbol{y}(obs)$ and $\boldsymbol{y}(calc)$ (in n-dimensional space) if scaling is done correctly and when its value is normalized by the length of the scaled observed data vector $\boldsymbol{y}'(obs)$, $|\boldsymbol{y}'(obs)| = (\sum_{i=1,\,n} y'_i(obs)^2)^{1/2}$.

In that case the *RMS* value ranges from 0 to 1 (0 meaning perfect agreement). The normalized version of expression [I] is given by:

$$RMS^{norm} = (\sum_{i=1,\,n} (y'_i(obs)-y_i(calc))^2)^{1/2} / (\sum_{i=1,\,n} y'_i(obs)^2)^{1/2} \qquad [II]$$

Scaling of $y(obs)$ with $s(obs)$ must be done according to the least squares protocol:

$$y'_i(obs) = s(obs)y_i(obs) \qquad [III]$$
$$s(obs) = \sum_{i=1,\,n} y_i(obs)\, y_i(calc) / \sum_{i=1,\,n} y_i(obs)^2 \qquad [IV]$$

As stated above, *RMS* is the length of the difference vector spanned by the observed and calculated data vectors but, due to scaling and normalization, this becomes the sine of the angle spanned by the observed and calculated data vectors $\boldsymbol{y}(obs)$ and $\boldsymbol{y}(calc)$: the difference vector is placed at an angle of 90 degrees onto the calculated data vector by scaling according to the L.S. protocol. Many times scaling is done incorrectly, *e.g.* on the basis of the total number of counts in the patterns. In that case the value of *RMS* is influenced not only by the angle between the n-dimensional vectors but also by scaling errors. Note that the order of the data points is unimportant for the calculation of *RMS*.

A well known criterion for powder pattern similarity is R_{wp} (R-weighted pattern). This criterion is used for the refinement of crystal structures on PXRD data and for structure solution from PXRD data. R_{wp} is similar to *RMS* but the squared differences are weighted according to the standard deviations in the observed intensities.

$$R_{wp} = (\sum_{i=1,\,n} w_i \, (y_i(obs) - y_i(calc))^2)^{1/2} / (\sum_{i=1,\,n} w_i \, y_i(obs)^2)^{1/2} \qquad [V]$$

$$w_i = 1/y_i(obs) \qquad [VI]$$

This weighting of the individual contributions comes from least squares theory. It can easily be shown that R_{wp} can be seen as a plain *RMS* criterion applied to modified powder patterns.

$$R_{wp} = (\sum_{i=1,\,n} (y_i(obs)^{1/2} - y_i(calc)/y_i(obs)^{1/2})^2)^{1/2} / (\sum_{i=1,\,n} y_i(obs))^{1/2} \qquad [VII]$$

For this modification the pattern values in both the observed and calculated patterns should be divided by the standard deviations in the observed intensities:

$$y^{mod}{}_i(obs) = y_i(obs)^{1/2} \qquad [VIII]$$

$$y^{mod}{}_i(calc) = y_i(calc)/y_i(obs)^{1/2} \qquad [IX]$$

$$R_{wp} = (\sum_{i=1,\,n} (y^{mod}{}_i(obs) - y^{mod}{}_i(calc))^2)^{1/2} / (\sum_{i=1,\,n} y^{mod}{}_i(obs)^2)^{1/2} \qquad [X]$$

As a result, also the scaling of the patterns is different from the one used for application of the *RMS* criterion.

$$s(obs) = (\sum_{i=1,\,n} y_i(calc)^2 / y_i(obs) / \sum_{i=1,\,n} y_i(obs))^{1/2} \qquad [XI]$$

The weighting used in R_{wp} is, however, only meaningful when the peak positions in the observed and calculated patterns are the same. The use of R_{wp} for the comparison of powder patterns corresponding to structures with (significantly) different unit cells is therefore incorrect.

The *WCC* (Weighted Cross Correlation) criterion[4] is based on correlation functions applied to unmodified patterns.

$$WCC = \int w_{fg}(r) c_{fg}(r) \mathrm{d}r \, / \, (\int w_{ff}(r) c_{ff}(r) \mathrm{d}r \int w_{gg}(r) c_{gg}(r) \mathrm{d}r)^{1/2} \qquad [XII]$$

$$c_{fg}(r) = \int f(x) g(x+r) \mathrm{d}x \qquad [XIII]$$

The weighting function, which extracts information from the correlation functions, can be adapted to influence the sensitivity for shifts in peak positions, in XRD as a results of lattice parameter variations. This is done by changing the value of l (values of l between 0.6 and 3.0 lead to stable and comparable results[4]; the IsoQuest program[5] uses 2.0):

$$w_{fg}(r) = 1 - |r| / l \quad \text{if } |r| < l \qquad [XIV]$$

$$w_{fg}(r) = 0 \quad \text{if } |r| \geq l \qquad [XV]$$

$$w_{ff}(r) = w_{gg}(r) = w_{fg}(r) \qquad [XVI]$$

The *WCC* criterion is equal to the cosine of the angle between two vectors in n-dimensional space when you only look at $r = 0$ (pointwise). In that case

$$WCC_{r=0} = (1 - RMS^2)^{1/2} \qquad [XVIII]$$

For $r > 0$ the comparison no longer holds.

The *WCC* criterion is always normalized and scaling is unnecessary since this is done implicitly. Comparison of deformed patterns, caused by unit cell variations, is possible with the *WCC* criterion since it recognizes shifted peaks. It is more sensitive to intensity variations since you look at unmodified patterns. However, this can easily be changed by mimicking the R_{wp} dampening of the intensities by taking the square root of the patterns (this is done in the program IsoQuest, which is discussed below). When we modify the patterns according to [VIII] and [IX] and calculate *WCC* at $r = 0$, we obtain the relation between *WCC* and R_{wp}, a relation between a sine and cosine function:

$$WCC^{mod}_{r=0} = (1 - R_{wp}^2)^{1/2} \qquad \text{[XIX]}$$

So, for $r = 0$ there is a direct and clear relation between *RMS*, R_{wp} and *WCC*.

From powder diffraction patterns to structural similarity

The Fortran code for calculating the match between two powder diffraction patterns, using weighted cross correlation functions, is remarkably simple. With NX being the number of data points, XS the step size in 2theta, WI the value for the width of the weighting function, and C1 and C2 arrays containing the intensity values for pattern 1 and 2, the similarity SI (ranging from 0 to 1, 1 meaning perfect agreement) can be calculated with the two subroutines MATCH and CORREL, given below.

In Appendix 1 additional code and data (Table 4 and 5) is given for test purposes. The necessary powder diffraction patterns and also the Fortran code can be obtained from http://www.crystallography.nl/xmatch/

.

.

```
      SUBROUTINE MATCH(C1,C2,NX,XS,WI,SI)
      DOUBLE PRECISION SI,A1,A2,CC
      INTEGER NX
      DIMENSION C1(NX),C2(NX)
      NW=NINT(WI/XS)
      RW=1./FLOAT(NW)
      CALL CORREL(C1,C1,NX,NW,RW,A1)
      CALL CORREL(C2,C2,NX,NW,RW,A2)
      CALL CORREL(C1,C2,NX,NW,RW,CC)
      SI=CC/SQRT(A1*A2)
      RETURN
      END
C
      SUBROUTINE CORREL(C1,C2,NX,NW,RW,CC)
      DIMENSION C1(NX),C2(NX)
      DOUBLE PRECISION T,CC
      CC=0.0
      DO 1 N=-NW+1,NW-1
        T=0.0
        DO 2 I=1,NX
          J=I+N
          IF(J.LT.1.OR.J.GT.NX) GOTO 2
          T=T+C1(I)*C2(J)
    2   CONTINUE
        CC=CC+T*(1.-RW*ABS(N))
    1 CONTINUE
      RETURN
      END
```

IsoQuest: application of crystal structure comparison to the CSD database

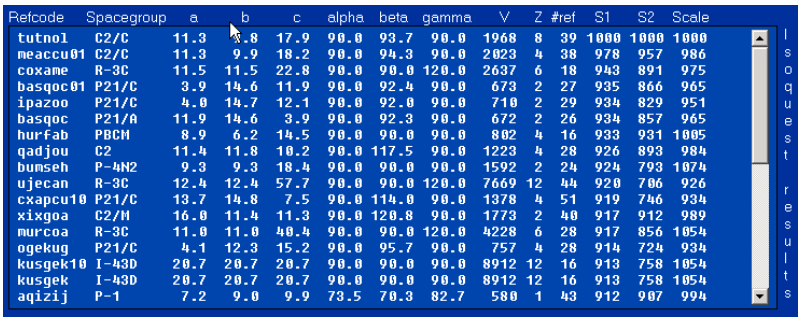


Fig 4: *Snapshot of the IsoQuest program showing that the structures TUTNOL and MEACCU01, initially seen as different structures, are redeterminations*

On the basis of the method described above we developed the program IsoQuest[5] for exploring the Cambridge Structural Database[1]. IsoQuest tells you whether there are structures that are identical or similar to your structure (see Fig. 4). Hits from IsoQuest can be further analyzed with IsoBase, a database containing the complete analysis of all (50 billion) structural relations in the CSD on the basis of the IsoQuest principle.

An in-house powder diffraction database was derived from CSD entries for which 3D-coordinates are available (2theta range from 0 to 40 degrees). For this purpose a program was written that can convert the complete CSD to a powder diffraction pattern database. The calculation of the PXRD patterns from the CSD entries is not straightforward. A lot of entries show disorder, contradictions in symmetry, non-existing elements, etc. Many of these problems can, however, be 'repaired'. The resulting powder pattern database can be searched by the IsoQuest program, using a structural model or reflection data as input.

With the IsoQuest program complete and often surprising lists of identical or related structures can be found in the CSD, in less than half a minute on a modern PC. Similar structures can be further explored with IsoBase, a database which contains all isostructurality relations in the CSD. IsoBase was generated by comparing all entries with all other entries in the CSD. The advantage of this database is that an IsoQuest search for a new structure can be further analyzed in a fast way. Why a further analysis? Structures found by IsoQuest can be related to other structures, initially not found during the first search, and this can be checked using IsoBase (see examples below) by taking automatically all IsoQuest hits as (secondary) search structures.

Example 1: analyzing compound "ALACAC01"

An IsoQuest search for compound ALACAC01 (an arbitrary entry from the CSD) gives the following result:

Table 1 IsoQuest result for compound ALACAC01

REFCODE	spacegroup	a	b	c	alpha	beta	gamma	volume	Z	nref	S1	S2	scale
ALACAC01	P21/C	14.1	7.6	16.4	90.0	99.0	90.0	1722	4	69	1000	1000	1000
ALACAC03	P21/C	14.0	7.5	16.3	90.0	98.8	90.0	1699	4	68	997	982	1005
ALACAC02	P21/C	14.0	7.5	16.3	90.0	98.9	90.0	1694	4	69	997	978	1005
ALACAC12	P21/C	14.0	7.5	16.3	90.0	98.8	90.0	1702	4	67	992	980	1005
ALACAC11	P21/C	14.0	7.5	16.3	90.0	98.8	90.0	1702	4	69	992	980	1005
ACACCR	P21/C	14.0	7.6	16.4	90.0	99.1	90.0	1714	4	69	988	987	1000
ACACCR02	P21/C	14.0	7.5	16.4	90.0	99.0	90.0	1709	4	69	986	984	1005
ACACMN02	P21/C	14.0	7.6	16.4	90.0	99.3	90.0	1721	4	68	985	985	1000
COACAC03	P21/C	13.9	7.5	16.2	90.0	98.4	90.0	1665	4	66	981	944	1010
QAMCAI	P21/C	14.2	7.6	16.6	90.0	99.3	90.0	1758	4	71	981	966	995
ACACMN	P21/C	13.9	7.5	16.2	90.0	98.4	90.0	1661	4	65	980	939	1010
COACAC02	P21/C	14.0	7.5	16.2	90.0	98.5	90.0	1672	4	65	975	945	1010
ACACGA	P21/C	14.0	7.6	16.5	90.0	99.0	90.0	1731	4	70	974	973	1000
ACACRU03	P21/C	14.0	7.5	16.3	90.0	99.0	90.0	1702	4	69	966	952	1005
ACACRU04	P21/C	13.8	7.4	16.1	90.0	99.3	90.0	1619	4	63	965	878	1020
ACACRH10	P21/C	13.9	7.5	16.4	90.0	98.6	90.0	1689	4	68	952	935	1010
KABMIJ	P21/C	14.1	7.5	16.5	90.0	99.0	90.0	1717	4	70	950	950	1000
ACACRU	P21/C	13.9	7.5	16.0	90.0	99.1	90.0	1650	4	63	949	913	1015

Using the result from IsoQuest and consulting IsoBase leads to five additional hits: NIRLAB, QQQCXJ01, QQQCXJ02, ACACRU05 and ACACRU06.

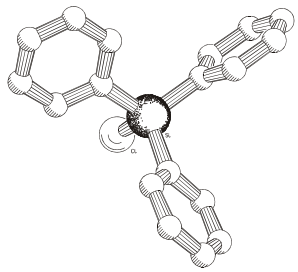
Table 2 IsoBase results for compound ALACAC01

REFCODE	spacegroup	a	b	c	alpha	beta	gamma	volume	Z	nref	S1	S2	scale
NIRLAB	P21/C	14.0	7.5	16.5	90.0	98.9	90.0	1707	4	69	963	961	1000
QQQCXJ02	P1121/B	13.9	16.4	7.5	90.0	90.0	98.6	1693	4	68	959	948	1005
QQQCXJ01	P1121/B	13.9	16.4	7.5	90.0	90.0	98.6	1693	4	68	959	948	1005
ACACRU06	P21/C	14.1	7.5	16.4	90.0	99.1	90.0	1713	4	69	951	949	1000
ACACRU05	P21/C	13.8	7.4	16.1	90.0	99.3	90.0	1638	4	63	949	879	1015

Refcodes which only differ in number (e.g. ALALAC01 and ALALAC03) correspond to the same compound. Therefore, the result of the combined use of IsoQuest and IsoBase yields 23 related entries, of which eleven entries are unique isostructural compounds.

Using the ligand system of entry ALACAC01 (attached to an "any"-atom) as a search fragment for the Conquest[6] program of the CCDC leads to only 14 of the 23 IsoQuest hits, of which 5 hits are unique isostructural compounds. This illustrates that entries can easily be missed and that IsoQuest and IsoBase generate additional information.

Example 2: analyzing compound "BARNUD"



A compound consisting of three phenyl rings and one Cl-atom attached to a central Si-atom, space group $P2_1/c$ (see picture[7] on the right) was used as input. With IsoQuest 16 related structures are found for this compound. The results for this search are given in Table 3.

Table 3. Results of a CSD-search for BARNUD using IsoQuest

CSD refcode	Chemical and Crystallographic similarity to BARNUD
WAKHOF	Cl replaced by SH
MTPHEP01	Cl replaced by (double bonded) CH_2
MTPHEP10	= MTPHEP01 but measured at a different temperature
TPPOSS	Cl replaced by double bonded S
TPPOSS02	= TPPOSS but measured at a different temperature
TPPOSS03	triclinic polymorph of TPPOSS (P-1)
TPGEBR	Si replaced by Ge, Cl replaced by Br
TPPHSE	Si replaced by P, Cl replaced by Se
TPPHSE01	triclinic polymorph of TPPHSE
TPASNS	Si replaced by As, Cl replaced by double bonded S
TPASNS01	same as TPASNS
BONLEV	Si replaced by P, Cl replaced by BH_3, spacegroup P-1
BRTPSN	Si replaced by Sn, Cl replaced by Br
TPSNCL01	Si replaced by Sn
TPSNCL02	= TPSNCL01 but measured at a different temperature
QUQCUA	Si replaced by Ge

In this set you find all kinds of replacements. Moreover, the unit cell or space group is not always the same. Different authors and journals are found for the same or similar compounds. Specifying a fragment in Conquest, consisting of three phenyl rings attached to an "any"-atom, and combining this fragment-search with a search on unit cell (standard tolerance of 1.5%) gives you 23 hits: BURNAD, BUFFAJ, CUTYAR, FONLAV, FXFPRP, GELCEF, HAFSIQ, HELYUS, JOZVUP, PZGACU, TPASNS, TPASNS01, TPGEBR, TPPHSE, TPSNCL02, WAKHOF, BAFLUQ, QUQCUA, NAKJUF, NAKJUF01, OGONIH, NAXFIC and WAWHOS. This set of 23 compounds contains 7 of the 16 isostructural compounds found by IsoQuest and 16 compounds that are not related to BARNUD in terms of isostructurality, showing again that IsoQuest generates additional information.

Example 4: analyzing an in-house crystal structure

Are there structures in the CSD that have similar packings as one of the structures published by our crystallography department in Nijmegen? A remarkable result is found for compound DEQGUB. The published crystal structures with CSD-refcodes WARDUO and NESDAQ are isomorphous although the cations in these structures are quite different (see Fig. 5). Apparently, the tetraphenylborate counterion (not shown) is dominating the crystal structures of these compounds.

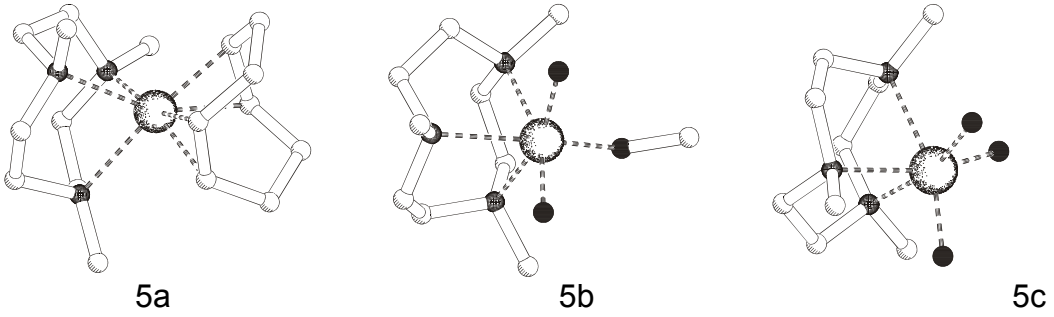


Fig 5: *Cations present in 3 isostructural compounds. a: DEQGUB, b: WARDUO, c: NESDAQ*

Example 5: large sets of isostructural compounds

IsoBase contains all isostructurality relations in the CSD and it can show for every entry the related compounds. There are many large sets of isostructural compounds visible in this database. An example is the set containing compound ECARAB. An amazing number of 42 isostructural compounds can be found for this cholic acid clathrate: ZUZDON, GUNKOP, GUNMUX, GUNLUW, BIFQAI01, LAFCEA, TEMWOX, GUNLOQ, ERIPUQ01, VABSOG, ERIPUQ, ZUZDUT, WEYNUJ, GUNNIM, GUNLAC, UMABAL, RABKEG, YUNYOV01, LAFCAW, PUWREE, ZUZFAB, YUNYOV, PIWKUB, LAFCAW01, GUNMAD, GOVRAK, MIWTIV, PIWKOV, PIWKOV01, LAHPOA, ZUPKIE, HURPEP, GUXFEK, HURNUD, ZEJFEZ, GUXFOU, GUXGAH, GUNMEH, GUXFAG, GOVQOX, VUTWAI and JOLFIZ.

Three of them (ECARAB, GOVRAK and HURPEP) are shown in Fig. 6. GOVRAK is a cholamide p-toluidine clathrate and consists of different host and guest molecules compared to ECARAB. HURPEP is also isostructural to ECARAB but has twice the unit cell volume.

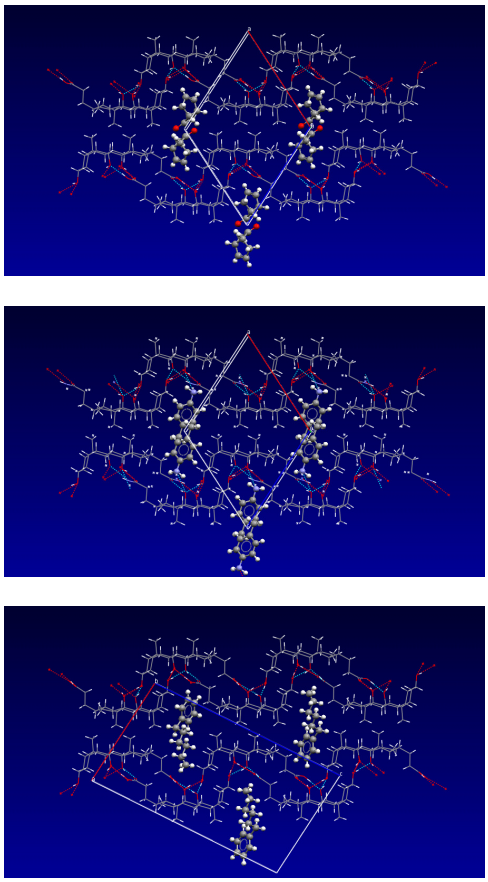


Fig 6: top: *ECARAB* **; middle:** *GOVRAK* ; **bottom:** *HURPEP*

This last example demonstrates again that the method described in this article is able to trace related compounds with differences in chemistry and/or symmetry and can collect sets of structures from the CSD database that can be used for studying the phenomenon of isostructurality or (pseudo)polymorphism.

Conclusions

The method for crystal structure comparison presented in this article is a powerful tool for crystallographers to check the uniqueness of newly determined structures, to look for related structures and to classify sets of structures. Moreover, applied to structural databases it is a way to generate new information on packing principles and crystal engineering rules.

References

1) The Cambridge Structural Database: a quarter of a million crystal structures and rising.
F. H. Allen, *Acta Crystallogr.*, **B58**, 380-388, 2002.
2) Method for the computational comparison of crystal structures. E.L. Willighagen, R. Wehrens, P. Verwer, R. de Gelder, L.M.C. Buydens, *Acta Crystallogr.*, **B61**, 29-36, 2005.
3) Mercury: visualization and analysis of crystal structures.
C. F. Macrae, P. R. Edgington, P. McCabe, E. Pidcock, G. P. Shields, R. Taylor, M. Towler and J. van de Streek, *J. Appl. Cryst.*, **39**, 453-457, 2006.
4) A Generalized Expression for the Similarity of Spectra: Application to Powder Diffraction Pattern Classification.
R. de Gelder, R. Wehrens and J.A. Hageman. *J. Comp. Chem.*, **22**, 273-289, 2006.
5) SYSTER and ISOQUEST: How good and unique are your data and structure?
R. de Gelder and J.M.M. Smits. *Acta Crystallogr.*, **A60**, s78, 2004.
6) New software for searching the Cambridge Structural Database and visualising crystal structures. I. J. Bruno, J. C. Cole, P. R. Edgington, M. Kessler, C. F. Macrae, P. McCabe, J. Pearson and R. Taylor, *Acta Crystallogr.*, **B58**, 389-397, 2002.
7) PLATON, A Multipurpose Crystallographic Tool. A.L. Spek, Utrecht University, Utrecht, The Netherlands, 2005.

Appendix 1:

```
      PROGRAM XMATCH
C
C  Last update: 191006 RdG
C
C  Calculates the match between powder diffraction patterns.
C  Returns a value between 0.0 and 1.0 for the similarity (SI).
C  1.0 means perfect agreement, 0.0 means completely different.
C
C  NP: number of powder patterns
C  NX: number of data points
C  XS: step size in 2theta
C  WI: single precision value for the width of the weighting function
C  SI: a single double precision value for the similarity
C  CO: intensity values for each pattern
C  CM: similarity matrix
C
      DOUBLE PRECISION SI
      DIMENSION CO(3001,20),CM(20,20)
      NP=20
      NX=3001
      XS=0.02
      WI=0.62
      OPEN (UNIT=2, FILE='wcc.out', STATUS='UNKNOWN')
      CALL READPR(CO,NP,NX)
      DO 1 I=1,NP
        DO 2 J=I,NP
          CALL MATCH(CO(1,I),CO(1,J),NX,XS,WI,SI)
          CM(I,J)=SI
          CM(J,I)=SI
    2   CONTINUE
    1 CONTINUE
      WRITE(2,4) WI
      WRITE(2,5) (I,I=1,NP)
      DO 3 I=1,NP
        WRITE(2,6) I, (CM(I,J),J=1,NP)
    3 CONTINUE
    4 FORMAT(' Width of the weighting function = ',F5.2)
    5 FORMAT(7X,1X,20(I5,1X))
    6 FORMAT(I7,1X,20(F5.2,1X))
      STOP
      END
C
      SUBROUTINE READPR(CO,NP,NX)
      DIMENSION CO(3001,20)
      CHARACTER*12 FN(20)
      DATA FN /'cefraa.dat','cefrab.dat','cefrna.dat','cefraq.dat',
     1         'cf2acn.dat','cfbipi.dat','ceflx2.dat','ceflan.dat',
     2         'cecloa.dat','second.dat','cfrob3.dat','copabe.dat',
     3         'codi26.dat','codi27.dat','cfpabe.dat','cef2ff.dat',
     4         'cfhchn.dat','cf4maf.dat','cfdmfs.dat','cfm3hb.dat'/
      WRITE(2,*)'******************************************************'
      WRITE(2,*)'Test : 20 powder patterns of cephalosporin complexes .'
      WRITE(2,*)'See: J. Comp. Chem. (2001), 22, 3, 273-289.'
      WRITE(2,*)'Results must agree with data in Table 5.'
      WRITE(2,*)'******************************************************'
      DO 1 I=1,NP
        OPEN (UNIT=1, FILE=FN(I), STATUS='OLD')
        DO 2 J=1,NX
          READ(1,*,END=3) X,CO(J,I)
    2   CONTINUE
    3   CLOSE (1)
        WRITE(2,4) I,FN(I)
    1 CONTINUE
    4 FORMAT(' Pattern ',I2,': ',A12,' read')
      WRITE(2,*)'******************************************************'
      RETURN
      END
```

Table 4 Crystal data corresponding to the test powder diffraction patterns

Compound name	a	b	c	α	β	γ	Space group	Type
Cefradine/alpha-naphthol	23.47079	7.12154	14.93036	90.0	108.26834	90.0	C2	A1
Cefradine/beta-naphthol	23.42119	6.97147	15.00473	90.0	110.40521	90.0	C2	A2
Cefradine/naphthalene	23.45837	7.11788	14.89216	90.0	108.57098	90.0	C2	A3
Cefradine/quinoline	23.41265	7.10910	14.80598	90.0	108.14590	90.0	C2	A4
Cefradine/2-hydroxy acetonaphthone	23.38549	7.19649	14.75882	90.0	108.57974	90.0	C2	A5
Cefradine/bipyridine	23.02269	7.14670	14.55443	90.0	104.64385	90.0	C2	A6
Cephalexine/beta-naphthol	23.39759	7.06229	14.91757	90.0	109.79842	90.0	C2	A7
Cephalexine/alpha-naphthol	23.43432	7.10804	14.87538	90.0	108.19072	90.0	C2	A8
Cefaclor/alpha-naphthol	23.48840	7.07855	14.84551	90.0	108.94678	90.0	C2	A9
Cefaclor/beta-naphthol	23.44692	7.02619	14.84134	90.0	110.55009	90.0	C2	A10
Cephadroxil/beta-naphthol	7.11174	21.71704	30.95857	90.0	90.0	90.0	$P2_12_12_1$	B11
Cephadroxil/4-hydroxy-benzoezuur	6.99921	20.99127	30.69011	90.0	90.0	90.0	$P2_12_12_1$	B12
Cephadroxil/2,6-dihydroxy-naphthalene	7.10786	21.86340	32.30589	90.0	90.0	90.0	$P2_12_12_1$	B13
Cephadroxil/2,7-dihydroxy-naphthalene	7.09018	21.27323	31.00436	90.0	90.0	90.0	$P2_12_12_1$	B14
Cefradine/4-hydroxy-benzoezuur	14.91661	7.38199	20.50296	90.0	105.77318	90.0	$P2_1$	C15
Cefradine/2-phenylphenol	23.56421	7.13203	18.68928	90.0	109.37986	90.0	C2	D16
Cefradine/hydroquinone	7.07185	10.70306	14.23422	87.15449	78.99942	89.74252	P1	E17
Cefradine/4-methylacetophenone	15.40382	7.29832	23.57355	90.0	99.35406	90.0	$P2_1$	F18
Cefradine/DMF	10.87473	9.51140	12.39035	90.0	98.70461	90.0	$P2_1$	N19
Cefradine/methyl 3-hydroxybenzoate	10.90731	9.40654	12.19924	90.0	98.53256	90.0	$P2_1$	N20

Table 5 Pattern similarity values for the test powder diffraction patterns

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	B11	B12	B13	B14	C15	D16	E17	F18	N19	N20
A1	1.00	0.80	0.99	0.96	0.93	0.88	0.93	0.99	0.92	0.82	0.78	0.60	0.78	0.72	0.75	0.80	0.63	0.66	0.34	0.41
A2		1.00	0.82	0.82	0.78	0.78	0.93	0.83	0.87	0.95	0.80	0.75	0.75	0.80	0.76	0.77	0.68	0.69	0.34	0.41
A3			1.00	0.98	0.92	0.90	0.94	0.99	0.93	0.83	0.81	0.63	0.79	0.75	0.76	0.82	0.66	0.67	0.35	0.41
A4				1.00	0.90	0.90	0.93	0.97	0.91	0.81	0.79	0.63	0.76	0.74	0.76	0.78	0.71	0.67	0.34	0.39
A5					1.00	0.82	0.86	0.91	0.85	0.79	0.72	0.61	0.75	0.69	0.81	0.75	0.58	0.73	0.31	0.40
A6						1.00	0.85	0.89	0.85	0.79	0.85	0.69	0.75	0.77	0.74	0.75	0.63	0.68	0.33	0.38
A7							1.00	0.95	0.96	0.94	0.81	0.66	0.77	0.78	0.75	0.81	0.66	0.66	0.35	0.40
A8								1.00	0.95	0.85	0.80	0.61	0.77	0.74	0.75	0.80	0.65	0.65	0.34	0.41
A9									1.00	0.93	0.79	0.66	0.75	0.77	0.71	0.78	0.63	0.64	0.34	0.39
A10										1.00	0.78	0.71	0.73	0.78	0.73	0.77	0.60	0.66	0.32	0.37
B11											1.00	0.84	0.87	0.96	0.73	0.83	0.66	0.69	0.41	0.42
B12												1.00	0.78	0.89	0.70	0.67	0.67	0.74	0.42	0.41
B13													1.00	0.87	0.77	0.83	0.66	0.75	0.42	0.44
B14														1.00	0.73	0.78	0.70	0.70	0.44	0.43
C15															1.00	0.67	0.70	0.75	0.43	0.44
D16																1.00	0.55	0.63	0.36	0.35
E17																	1.00	0.60	0.37	0.45
F18																		1.00	0.42	0.53
N19																			1.00	0.80
N20																				1.00

Topological analysis of crystal structures

Oleg V. Dolomanov
Department of Chemistry, The University of Durham, South Road, Durham, DH1 3LE, UK. Correspondence e-mail: pcxod@nottingham.ac.uk ; WWW: http://www.ccp14.ac.uk/ccp/web-mirrors/lcells/

Introduction

Extended structural networks are present in a wide range of chemical compounds. The importance of analysis of these networks stems from the need to classify the materials in terms of their three-dimensional structure and the construction of different structure-property correlations. The classification assumes the existence of an *invariant* which consistently describes the framework of these materials. A number of invariants have been introduced for the description of extended frameworks, such as

- Schläfli symbols, vertex symbols or topological terms [1]
- Coordination sequences and topological density [2-4]
- Cycle sequences [5, 6]

These invariants have been mainly used for zeolitic frameworks, which consist only of tetrahedral nodes. However, Schläfli symbols in particular are finding increasing application in structural networks of different connectivity.

The easiest invariant to interpret is the Schläfli symbol, which describe the environment of a selected node within an extended network. Coordination sequences are not as easily interpreted as they describe summarised growth of nodes in coordination shells; being periodic, they can have a very long period, which can make them hard to use. The cycle sequences represent a combined approach, which does not make them as comprehensive as vertex symbols or more practical than coordination sequences. However, in some cases it is necessary to use a combination of these invariants to describe networks, in particular it is necessary for polymorphs (e.g. diamond and lonsdalite), because a single invariant may be not unique for highly related systems.

Evaluation of these characteristics for structural networks is straightforward in the case of 2D structures, but may be very complicated in case of 3D structures. This article focuses on the algorithms for the construction of topological networks and evaluation of vertex symbols, which are implemented in the crystal structure visualisation and analysis program “Olex” [7].

Definitions

A *topological network* is the mathematical abstraction of the connectivity of an extended structural network. Such a network can be mono- or multi-component. The transformation of a chemical network into its topological representation proceeds as follows: if one of these entities (e.g. metal cation, ligand, coordinating solvent or counteranion) is connected to only one another, then it is removed; if it is connected to two others, it is replaced with a bond between the neighbouring entities; if it is linked to more than two others it is considered as a *node*. In this representation the network appears as a mathematical graph where each node is associated with a list of at least three other nodes. The *length of the pathway* between any two nodes is given by the number of nodes along it, including the first and the last nodes.

The definition of *a ring* which is used in this discussion is consistent with the definition of the *primitive ring* by Marians [8] and represents a circuit, which cannot be split into two smaller ones. Alternatively, it can be defined as a circuit, where all the shortest paths between any two nodes belong to the ring. These rings are also called *minimal rings* [9].

Term "*coordination shell*" is closely related to the term *coordination sequence* [2-4], and describe nodes topologically equidistant[8] from a selected one. Thus, the first coordination shell represents the immediate environment of a central node, whereas the second consists of nodes directly connected to the nodes of the first shell, but excluding the central atom. Coordination sequences and shells are related so that the k*th* member of the coordination sequence represents the number of atoms in the k*th* coordination shell.

An example

One of the simplest examples of extended network can be a square grid (Fig 1). To evaluate the vertex symbol for central node, unique circuits based on outgoing bonds have to be counted. The four four-membered circuits can be easily seen. These circuits also comply with the definition of ring. However, for a four- connected network 6 circuits is expected.[9] The other two six-membered circuits to be found are illustrated in Fig 1.

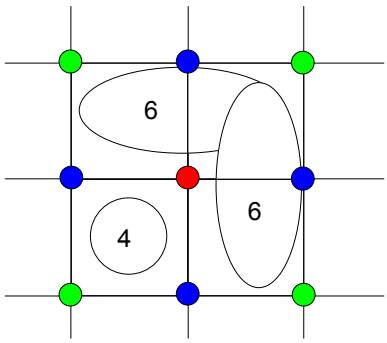


Figure 1: *View of an element of a square grid. The red node is central, blue nodes – the first coordination shell, green nodes – second coordination shell.*

These circuits are not rings, as they consist of two smaller four-membered rings and therefore do not comply with the ring definition. The composition of the vertex symbols will be discussed later. However, from this example it is seen that in order to evaluate vertex symbols the construction of topological network is required and an algorithm to find circuits formed by two bonds outgoing from chosen atom is needed as well.

Construction of topological networks

Extended structural networks can be very complicated due to the presence of a variety of chemical components and in this way differ from networks of inorganic zeolites, where structural and topological networks are almost identical. There are two approaches to the construction of topological networks: one is based on the application of the definition of a topological node (*mathematical abstraction*, Fig 2), while the other uses information about building blocks of the structure. The first algorithm does not require any user input, whereas the second one needs the user to define components of the structure. Despite being generic, the first algorithm may lead to topological networks with no chemical sense, and further intervention by the user is needed to interpret the resulting topological network in terms of the chemical species present (e.g. to identify which nodes belong to the ligand).

[8] Topological distance is counted in number links between nodes and therefore length of the pathway from these nodes to the selected one is equal.

[9] For an n-connected node the number of circuits to be found is n*(n-1)/2

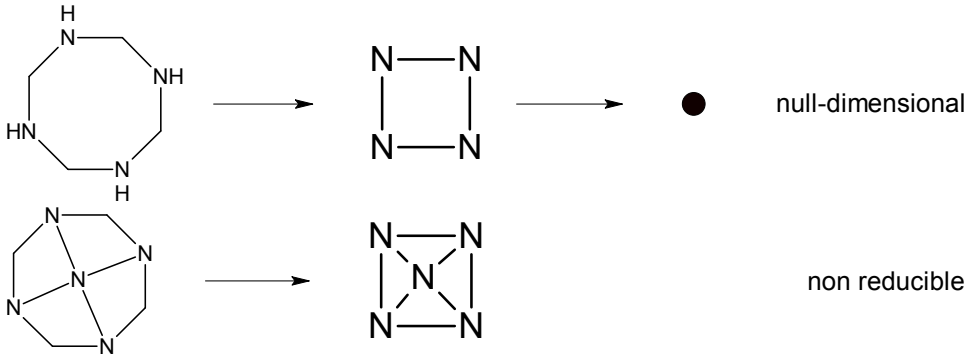


Fig 2: *View of the mathematical abstraction of a chemical structure. According to the described algorithm hydrogen atoms are removed and atoms connected to only two others are replaced with a bond.*

The mathematical abstraction algorithm for the construction of topological networks can be illustrated by the following code snippet:

Construction of topological network using mathematical abstraction approach snippet
The first approximation is that all atoms of the structure are nodes. Upon return, only nodes by definition will be left

```
nodesRemoved = 1
while (nodeRemoved)
{
  nodesRemoved=0;
  for (i=0; i<nodeCount; i++)
  {
    if (Node[i].NeighbourCount < 2)                          -node connected to one other
    {
      Node[i].Remove();                                      -remove the node
      i- -;
      nodesRemoved++;
      continue;
    }
    if( Node[i].NeighbourCount == 2)                         -linking node
    {
      Node[i].Neighbour[0].LinkTo( Node[i].Neightbour[1] );  -link the neighbouring nodes, remove current
      Node[i].Remove();
      i--;
      nodesRemoved++;
      continue;
    }
  }
}
```

Ring statistics in chemistry

Evaluation of vertex symbols requires counting the number of rings in the environment of a particular topological node. This is closely related to the concept of ring statistics, which has a very important role in chemistry. The ring system of a molecule is a key feature in determining its shape and properties. Ring statistics are also used for the interpretation of chemical structure, design of organic synthesis and storage of chemical information [10, 11]. Many computer algorithms have been developed to obtain specific information from chemical structures in terms of their ring interconnectivity for either discrete or polymeric structures. These algorithms, which were reviewed by Downs [12], can be divided into following groups:

- Graph theory
  general, spanning tree, edge combination, incidents matrix, connection table, adjacency matrix, half-ring tracing, breadth-first growth of spanning tree, maximal adjacency matrix
- Planar projection
- Walk through the connectivity table

There is a large choice of algorithms. Each was designed to perform a specific role, and many of the 24 algorithms, reviewed by Downs fail for complex ring systems. However, the description of topological networks using vertex symbols converges to a relatively simple task – the algorithm does not have to

evaluate all rings in the system; instead the rings have to be found in the environment of a selected node, which simplifies the job significantly.

Circuit perception

Yuan and Cormack [13] describe a convenient algorithm for primitive ring statistics in compounds with topological networks with intermediate connectivity (e.g. 4-connected), but without translational symmetry such as silicate glasses. The algorithm has several good points:

(1) absence of upper limit on ring size, (2) low memory usage, (3) speed

The algorithm consists of three subroutines: shortest path search, shortest path identification, and identification of primitive rings. Additionally the algorithm is optimised for the shortest-path search by specifying the search direction ("four-point directing single-pair search"), which gives it some advantages in calculation speed.

The algorithm, described here, has similar characteristics to the one above, but with emphasis on the evaluation of vertex symbols. It intensively utilises the modified breadth-first algorithm [14], which is used for rings perception as well as to find shortest path length between any two nodes of the network.[10] It was modified in order to find a circuit or a path between nodes; normally two nodes are used instead of a single node in the original algorithm. This modification gives good improvement in performance leading to about fourfold increase in speed as the volume of a sphere grows as its radius cubed (Fig 3).

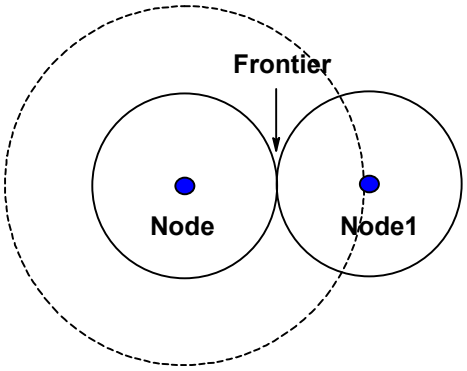


Fig 3: *An illustration of the modified breadth-first algorithm. The dotted line shows the volume of nodes which have to be processed when the algorithm is initiated from one node and the solid lines show the volume of nodes when the algorithm is initiated from both nodes simultaneously.*

It was found that simple breadth-first algorithm is not applicable to find all possible rings in the network due to the algorithm's anisotropic nature (Fig 4b,c), which appears in different node numbering in dependence on the starting node. Also it is almost impossible to use the *down-walk*[11] approach to find all possible rings around central node as illustrated in Fig 4a.

[10] Originally, this algorithm is used to build spanning trees for a graph; here it is used to assign atoms coordination shell numbers (tags), which further can be processed to retrieve necessary information.

[11] A walk between linked nodes from the one with higher tag to a smaller one; the difference between the tags must be 1.

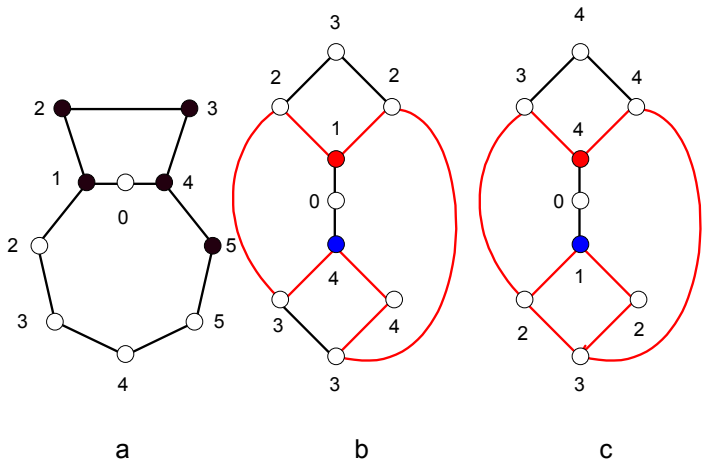


Fig 4: *(a) An illustration of a situation where the breadth-first algorithm fails - the longer 8-membered ring is not found. (b, c) An illustration of the anisotropic nature of the down-walk algorithm. The algorithm produces different numbering in dependence on the starting node.*

To eliminate these features of the algorithm, it was modified so that the numbering starts from both nodes simultaneously using different tags. Nodes, where the numbering conflict occurs (one tag is assigned a value and the other tag is about to be assigned) are remembered and the tag-assignment is halted. This modification is referred to as *the frontier search algorithm*. When the numbering stopped, the network consists of two hemispheres, where nodes have only one tag initialized out of two. Following completeness of the initialization from each conflicting node allows acquiring of all possible circuits using the down-walk approach.

The frontier search algorithm is illustrated in Fig 5. In Fig 5a the hemispheres are constructed and in Fig 5b the completeness of the initialization from the conflict nodes, shown in black, is illustrated. After the tags of nodes are initialised completely, the graph is ready for down-walk search of circuits (4a down to 1a in Fig 5b and 7a down to 1a in Fig 5c).

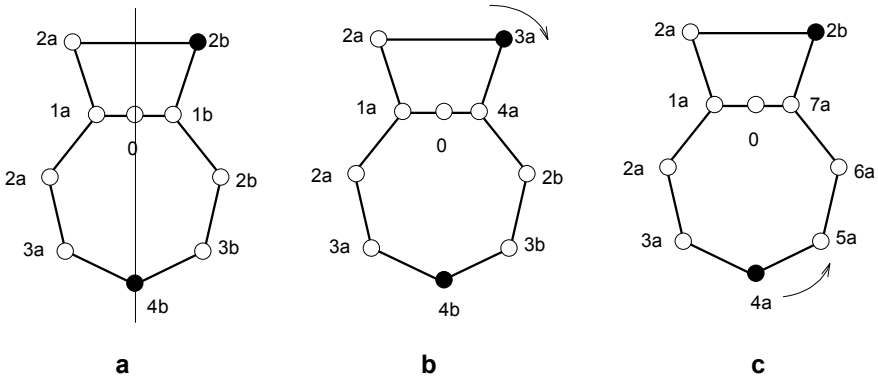


Fig 5: *An illustration of the frontier search algorithm. (a) View of tags after the execution of frontier search algorithm; nodes 4b and 2b are remembered. (b, c) Following completing initialisation of the "a" tags from the conflicting nodes allows finding either ring.*

The number of the conflict nodes is not very large in comparison with whole network (a plane in 3D), but the number of operations increases dramatically. To improve the performance, the conflict nodes have to be sorted by their tags in ascending order and the search for a primitive ring should be performed for each conflict node. If such a ring is found, the procedure has to be performed for the remaining of conflict nodes having the same number. This allows finding of all alternative equally-sized rings, as required for the evaluation of long vertex symbols.

The frontier search algorithm snippet.

This algorithm assumes that a node has two tags (can be implemented with two arrays for exiting code) and these tags are initialised with negative values (e.g. -1). NodeA and nodeB are the two nodes in the environment of the central node for which the frontier is to be found.

```
List shellA, shellB, frontier;
shellA.add( nodeA );                                  -initialise the first step
shellB.add( nodeB );
nodeB.Tag2 = 1;                                       -this is used to tackle three-membered rings
centraNode.Tag1 = centralNode.Tag2 = 0;               -this eliminates the central node from counting
while (true)                                          -infinite loop over all nodes
{
  length = shellA.length;                             -make sure that only one coordination shell is
  for (i=0; i<length; i++)                            processed a time
  {
    for (j=0; j<shellA[i].NeighbourCount; j++)
    {
      if ( shellA[i].Neighbour[j].Tag1 < 0 )          -remember the node – numbering “collision” oc-
      {                                               curred
        if (shellA[i].Neighbour[j].Tag2 > 0 )
          frontier.Add( shellA[i].Neightbour[j] );
        shellA[i].Neightbour[j].Tag 1= shellA[i].Tag1 + 1;   -assign next coordination number
      }
      shellA.Remove(0);                               -get rid of the processed nodes; different methods
      i--;                                            to improve performance can be implemented
    }
  }
  length = shellB.length;                             -make sure that only one coordination shell is
  for (i=0; i< length; i++)                           processed a time
  {
    for (j=0; j<shellB[i].NeighbourCount; j++)
    {
      if (shellB[i].Neighbour[j].Tag1 < 0  &&  shellB[i].Neighbour[j].Tag2 < 0   -do not have to search for the frontier here – the
)                                                     previous piece of code does it
        shellB[i].Neightbour[j].Tag2 = shellB[i].Tag2 + 1;
    }
    shellB.Remove(0);
    i--;
  }
  if( shellA.Empty && shellB.Empty )  break;          all nodes processed – may quit now; different
}                                                     conditions can be used to stop the process (e.g.
                                                      maximum circuit length)
```

The circuit perception algorithm snippet for two nodes in the immediate environment of the given node.

The algorithm assumes that node tags have been initialised as above. Circuits are evaluated for the central node. To evaluate all circuits n*(n-1)/2 iterations have to be done for all nodes in the environment of the central one.

```
List FoundCycles;                                     -this will be returned
ringLevel = 0;
ringFound = false;
SortFrontiedByTag1();                                 -sort all where numbering collision occurred by Tag1
RememberTags();                                       -have to remember the tags as they get changed later
for (i=0; i<frontier.length; i++)
{
  if( ringFound && frontier[i].Tag1 != ringLevel )  break;   -condition when all circuits of the same size are processed
  frontier[i].CompleteTag1Initialisation();           -complete initialisation of Tag1 for all nodes where it -1
  List circuits = frontier[i].WalkDownExpandCircuits();   -get the list of all circuits using down-walk approach
  for (j=0; j<circuits.length; j++)
  {
    if (circuit[i][ circuit.length-1] != nodeA )      -check that the circuit leads to nodeA
    {
      circuits.Delete(j);
      j--;
      continue;
    }
    if( !ShortcutExists( circuits[j] ) )              -check if current circuit is a ring; this will be one of conditions
    {                                                 to break further iterations
      ringFound = true;
      ringLevel = frontier[i].Tag1;
    }
  }
  FoundCycles.AddUnique( circuits );                  -add only unique circuits only (with different nodes in the
  RestoreTags();                                      middle)
}                                                     -restore tags for the next iteration
```

CompleteTag1Initialisation function snippet

The function assumes that node tags have been initialised by the frontier search algorithm	

for (i=0; i<**this**.NeigbourCount; i++)
{
  if(**this**.Neightbour[i].Tag1 < 0)
  {
  if(**this**.Tag2 < **this**.Neighbour[i].Tag2)
  {
  this.Neighbour[i].Tag1 = **this**.Tag1 + 1;
  this.Neghbour[i].CompleteTag1Initialisation();
  }
  }
}
	-iterate through all connected nodes
	
	-check if Tag1 is initialised
	
	-the down-walk condition
	
	-recursive call

The down-walk algorithm snippet These two function are used to list all circuits of a given node	

List WalkDownExpandCirrcuits()
{
  List circuits, circuit;
  this.WalkDown(circuit, circuits);
  return circuits;
}
WalkDown(List curerrentCircuit, List circuits)
{
  for (i=0; i<this.NeighbourCount; i++)
  {
  if((node.Tag1 – node.Neighbour[i]) == 1)
  {
  List newCircuit;
  newCircuit.Assign(currentCurcuit);
  circuitCurcuit.Add(node);
  node.Neighbour[i].WalkDown(newCircuit);
  circuits.Add(newCircuit);
  }
  }
}
	-a convenience function
	
	-walk down condition – the tags difference is 1
	
	-copy values from previous circuit segment so that the circuits consist of all nodes
	-recursive call

Circuits Analysis

We have tried different methods for the analysis of circuits, but no easily-implemented or reliable algorithms were found, and so full circuit analysis is performed. The full analysis is done by trying to find a shorter path between any two nodes of a circuit than the ones, which belong to the circuit. As mentioned above the shortest path length is evaluated using modified breadth-first algorithm.

Shortcut search algorithm snippet The algorithm assumes that content of the circuit is ordered sequentially, so that node index represent its distance from nodeA or nodeB	

boolean ShortcutExists(circuit) {
  for (i=0; i<circuit.length; i++)
  {
  for (j=i+2; j<circuit.length; j++)
  {
  length = circuit.Node[i].DistanceTo(circuit.Node[j]);
  if ((length < (j-i)) **&&**
  (length < (circuit.length +1 – (j-i))))
  return true;
  }
  }
  return false;
}
	-go through all nodes of the circuit
	
	i+2: nodes must be separated by at least two others for shortcut to exist
	
	-compare with one side of the circuit
	-compare with other side of the curcuit
	-shortcut found in this case
	
	-no shortcut found

Publishing symbols

The circuit search algorithm has to be applied to all unique pairs of bonds outgoing from the central atom. After this is done, vertex symbols can be easily evaluated. To evaluate the short vertex symbol, rings of different sizes formed by all distinct angles (an angle is formed by two bonds outgoing from the central node) are counted. If a ring does not exist, then a shortest circuit is taken according to Smith [15]; according to Wells [1] the shortest circuit is taken in any case. Then the rings are sorted in ascending size and the ring number is used as a superscript. It is noteworthy that some authors prefer to omit a circuit if the ring cannot be found. Thus for the network discussed in the example (Fig. 1) the short vertex symbol is $4^4 6^2$ (some authors prefer to refer to it as to the 4^4 grid).

Long vertex symbols are used for four-connected networks and can be composed by the following procedure described by O'Keeffe [16, 17]:

1. evaluate all rings (including alternative of the same size) for all six angles; if no rings can be found, the infinity sign (∞) is used
2. group symbols for opposing angles together
3. indicate the number of rings of the same size by a subscript, omitting one
4. order the rings in ascending order, considering point 2

Performing this procedure for the network in Fig 1 gives 4.4.4.4. ∞.∞.

Implementation

Algorithms described above are implemented in the Olex [7] program, which is freely available at the CCP14 website. Users can use Olex to open SHELX files, grow the structure using symmetry operations, construct a topological network and evaluate short/long vertex symbols for the structure. An interface is provided for the analysis of networks with short interactions (e.g. hydrogen bonds) and for assigning nodes of the automatically generated topological network to distinct chemical components of the network (e.g. ligand, counteranion, etc),

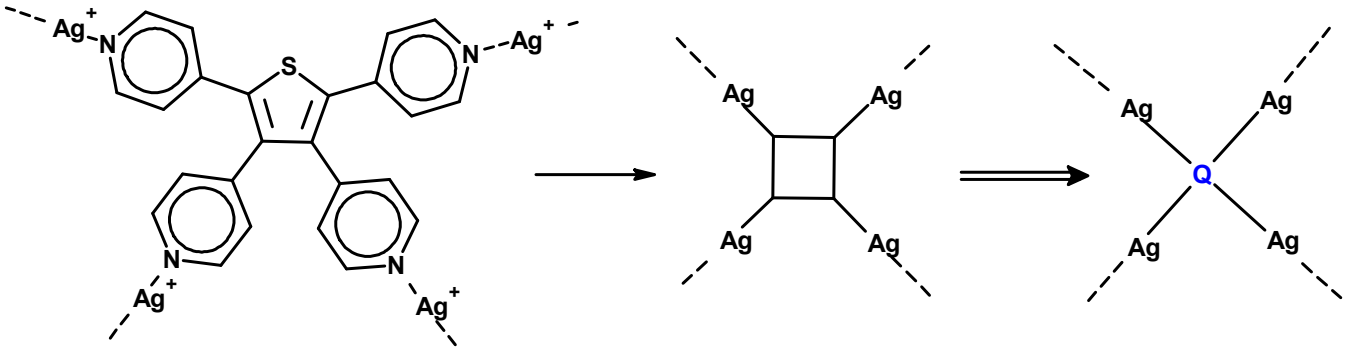


Figure 6: *An illustration of the topological network construction. The first step is the mathematical abstraction of the chemical network, the second step is to give the network a chemical sense.*

The program works with networks based upon covalent bonds and to analyse networks with short interactions the user can transform a list of selected short interactions into "covalent" bonds and can perform the analysis in the same way as for any other covalent network. The tool for making chemically sensible topological networks allows on to choose some topological nodes/bonds and to replace them with a new topological node (Fig 6, [18]).

Conclusions

Algorithms for topological analysis of extended structural networks and evaluation of vertex symbols are discussed and implementation is provided in code snippets. The algorithms were implemented in the "Olex" software and were tested on a variety of zeolitic and metalloorganic networks and had shown results consistent with previously published data.

Acknowledgements

I thank Dr Simon Parsons for help in preparation of this article.

References

1. A. F. Wells, 'Three-Dimensional Nets and Polyhedra', John Wiley & Sons, 1977
2. W. M. Meier and H. J. Möck, *J. Solid State Chem.,* 1979, **27**, 349-355
3. G. O. Brunner, *J. Solid State Chem.,* 1979, **29**, 41-45
4. R. W. Grosse-Kunstleve, G. O. Brunner and N. J. A. Sloane, *Acta Cryst.,* 1996, **A52**, 879-889
5. A. Beukemann and W. E. Klee, *Z. Kristallogr.,* 1994, **209**, 709
6. G. Thimm and W. E. Klee, *Zeolites,* 1997, **19**, 422-424
7. O. V. Dolomanov, A. J. Blake, N. R. Champness, C. Wilson and M. Schröder, *J. Appl. Cryst.,* 2003, **36**, 1283-1284
8. C. S. Marians and L. W. Hobbs, *J. Non-Cryst. Solids,* 1990, **124**, 242-253
9. L. Guttman, *J. Non-Cryst. Solids,* 1990, **116**, 145-147
10. Y. Takahashi, *J. Chem. Inf. Comput. Sci.,* 1994, **34**, 167-170
11. B. Schmidt and J. Fleischhauer, *J. Chem. Inf. Comput. Sci.,* 1978, **18**, 204-206
12. G. M. Downs, V. J. Gillet, J. D. Holiday and M. F. Lynch, *J. Chem. Inf. Comput. Sci.,* 1989, **29**, 172-187
13. X. Yuan and A. N. Cormack, *Computational Materials Sciences,* 2002, **24**, 343-360
14. T. H. Cormen, C. E. Leiserson and R. L. Rivest, 'Introduction to Algorithms', MIT Press, 1990
15. J. V. Smith, *American Mineralogist,* 1978, **63**, 960-969
16. M. O'Keeffe and B. G. Hyde, 'Crystal Structures I: Patterns and Symmetry', Mineralogical Society of America Monograph, 1996
17. M. O'Keeffe and S. T. Hyde, *Zeolites,* 1997, **19**, 370-374
18. O. V. Dolomanov, D. B. Cordes, N. R. Champness, A. J. Blake, L. R. Hanton, G. B. Jameson, M. Schröder and C. Wilson, *Chemical Communications,* 2004, 642-643

On the Detection of Solvent Accessible Voids in Crystal Structures with PLATON/SOLV

Anthony (Ton) L. Spek
Crystal & Structural Chemistry, Bijvoet Centre for Biomolecular Research, Science Faculty, Padualaan 8, Utrecht University, 3584 CH Utrecht, The Netherlands. Email: a.l.spek@chem.uu.nl ; WWW: http://www.cryst.chem.uu.nl/

1. Abstract

A technique is described, as implemented in the program PLATON, for the determination of solvent accessible voids in a crystal structure. Current applications of this utility include the SQUEEZE method for the handling of disordered solvents in crystal structure refinement, crystal structure validation with CHECKCIF to report on possibly missed voids in a structure model and the classification of pores in framework structures.

2. Introduction

Our interest in solvent accessible voids in a crystal structure stems from a problem that we encountered with the refinement of the crystal structure of the drug Salozopyrin that appeared to include continuous solvent channels [1]. The residual electron density in those channels (shown in Fig. 1) could not be modeled satisfactorily with a disorder model. The difference Fourier map (Fig. 2) showed no isolated peaks but rather a continuous density tube filled with unknown solvent. After having surmounted the problems to obtain suitable crystals for data collection we got subsequently stuck with an unsatisfactorily high and un-publishable R-value due to the unaccounted for solvent contribution. This problem was the start of the development of a technique now named SQUEEZE and available in the program package PLATON [2]. A description of a prototype implementation of the current method, at that time named BYPASS and interfaced with SHELX76, can be found in reference [3]. The underlying concept of SQUEEZE is to split the total scattering factor F_H(calc) into two contributions: the contribution of the ordered part F_H(Model) and the contribution of the disordered solvent F_H(solvent). The latter contribution is obtained by back-Fourier transformation of the electron density that is found in the solvent region in the difference Fourier map. This recovery procedure is repeated until conversion is reached. SQUEEZE interfaces with SHELXL97 [Göttingen] and CRYSTALS [Oxford].

Judging from the number of structures that are flagged in the CSD for the fact that SQUEEZE was used, it can be estimated that this procedure was used at least 1000 times and probably many times more.

This contribution will address only the algorithm that is used to establish the solvent accessible volumes in a crystal structure.

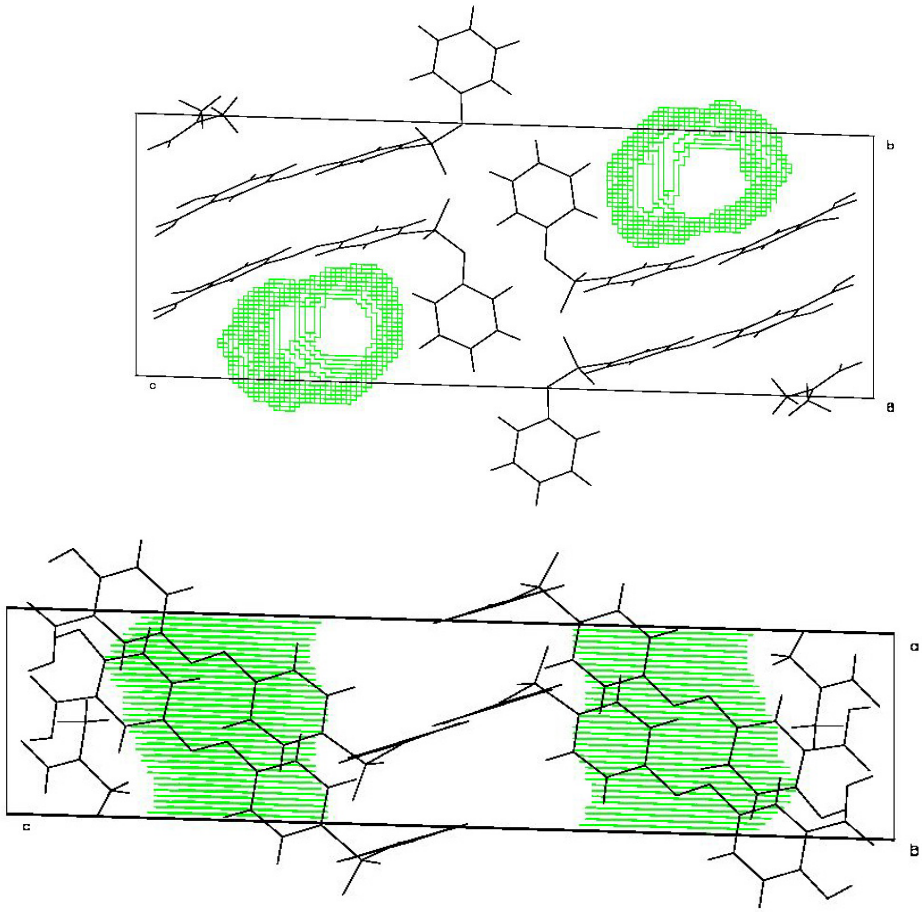


Figure 1: *Views down and perpendicular to the solvent accessible channels (green) in the crystal structure of the drug Salazopyrin*.

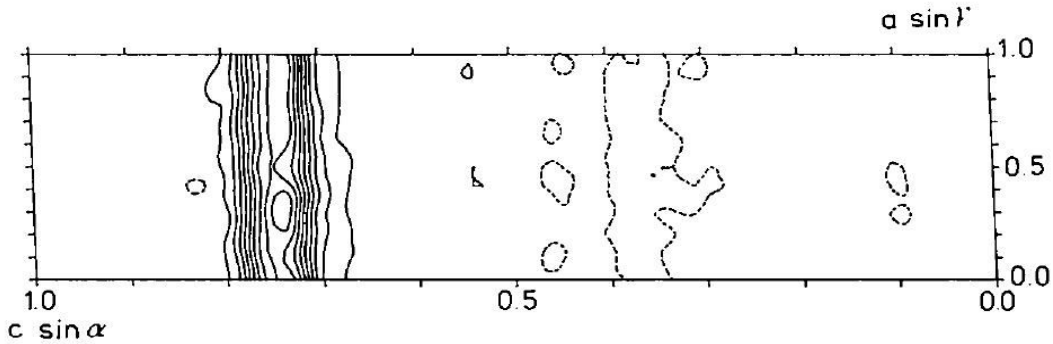


Figure 2: *Section through the difference Fourier map showing the content of the channels.*

Alternative approaches for the determination of voids have been reported by Richards [4], Ho & Marshall [5], Mugnoli [6], Watkin [7] and Wigderson & Meyer [8].

3. The Algorithm

All crystal structures contain void space in small regions and cusps between atoms. In the order of 35% of the space in a crystal structure lies outside the van der Waals volume of the atoms in the structure. In the current context, we are not interested in those voids but rather in voids that can at least accommodate a sphere with minimum radius R(min). A good default choice for R(min) = 1.2 Å, being the van der Waals radius of a hydrogen atom. Most structures exhibit no voids in the last sense.

In this section, we will first give an 'analog' graphical introduction to the concept of 'solvent accessible volume' and then more details on its numerical implementation of the concept.

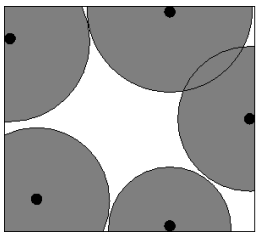
3.1 The analog model

Solvent accessible voids are determined in three steps as illustrated in fig. 3.

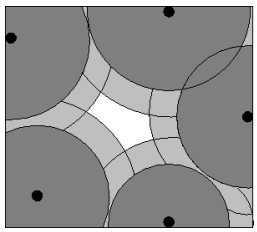
Step #1: A van der Waals radius is assigned to all atoms in the unit cell. In this way, we have divided the total volume V into two parts: V(inside) and V(outside). Note that as a byproduct, we can determine the so called Kitajkorodskij packing index [9]:
Packing Index = V(inside) / V.

Step #2: Define the volume within which all points are at least 1.2 Å away from the nearest van der Waals surface. This volume is obtained in a way similar to that in step #1 but now with atom radii being the van der Waals radii increased by 1.2 Å.

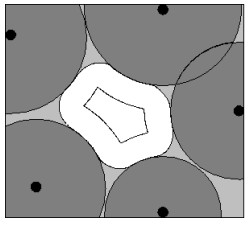
Step #3: Extend the volume obtained in step #2 with all points that are within 1.2 Å from its bounding surface.



STEP #1 – EXCLUDE VOLUME INSIDE THE VAN DER WAALS SPHERE



STEP # 2 – EXCLUDE AN ACCESS RADIAL VOLUME TO FIND THE LOCATION OF ATOMS WITH THEIR CENTRE AT LEAST 1.2 ANGSTROM AWAY



STEP # 3 – EXTEND INNER VOLUME WITH POINTS WITHIN 1.2 ANGSTROM FROM ITS OUTER BOUNDS

Figure 3: *Cartoons illustrating the three stages of the identification of the solvent accessible volume.*

3.2 The Numerical Implementation

The numerical implementation of the model described in section 3.1 is relatively compute intense. We give here only a sketchy description. The actual implementation involves significant bookkeeping and is best gleaned from the Fortran source. The calculations are based on a grid with a distance between gridpoints in the order of 0.2 Å. The exact value depends on the need to have an integral number of grid steps in each of the three dimensions.

Step #1: Each gridpoint is marked as either (I) inside the van der Waals volume, (II) at least 1.2 Å away from the nearest van der Waals surface or (III) neither (I) or (II). Note: for the SQUEEZE application, a faster shortcut is used but this will not be discussed here.

Step #2: Connected sets of gridpoints of type (II) are assembled into void objects (either as isolated entities or 1, 2 or 3 dimensional channels or networks.

Step #3: Each void object is extended with the gridpoints that are within 1.2 Å of the bounding surface.

For each void object, the volume and center of gravity is calculated (see Fig. 4). In addition a second moment of the distribution of the gridpoints is calculated and analysed in terms of main axes and variances. The latter gives some idea about the shape of the voids. In addition, the distance from the center of gravity to the nearest atoms is listed. This can be helpful in the interpretation of the type of void at hand. I.e. a void with a volume of in the order of 40 $Å^3$ with nearby acceptor atoms such as O or N can likely host a water molecule.

The complete void map can be inspected in printed sections or graphically (either in mono or stereo). Fig. 5 gives an example in mono style.

4. Applications

A major application of the algorithm as implemented in PLATON is its use as part of the SQUEEZE procedure. It is used as a mask on a difference Fourier map to extract relevant information on its contents such as the integrated number of electrons in the masked volume.

The example (Fig. 4) below concerns a void containing a THF molecule disordered over the two-fold axis in space group C2/c. The reported volume of 156 $Å^3$ is typical for such small molecules. The number in [] behind this value represents the type (II) volume.

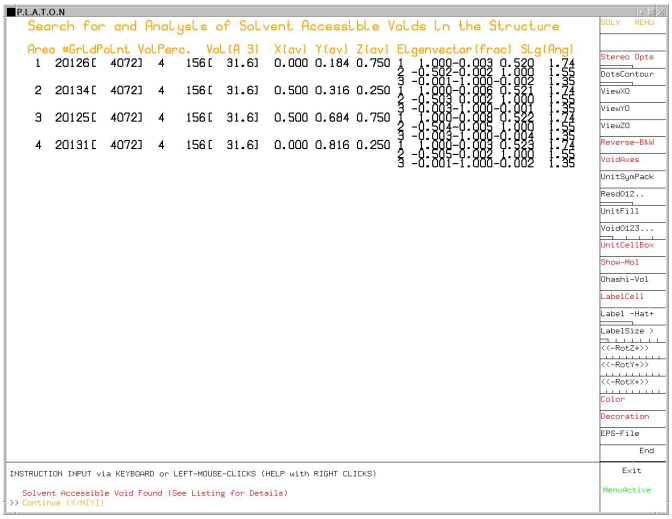


Figure 4: *Numerical display of the voids in the unitcell.*

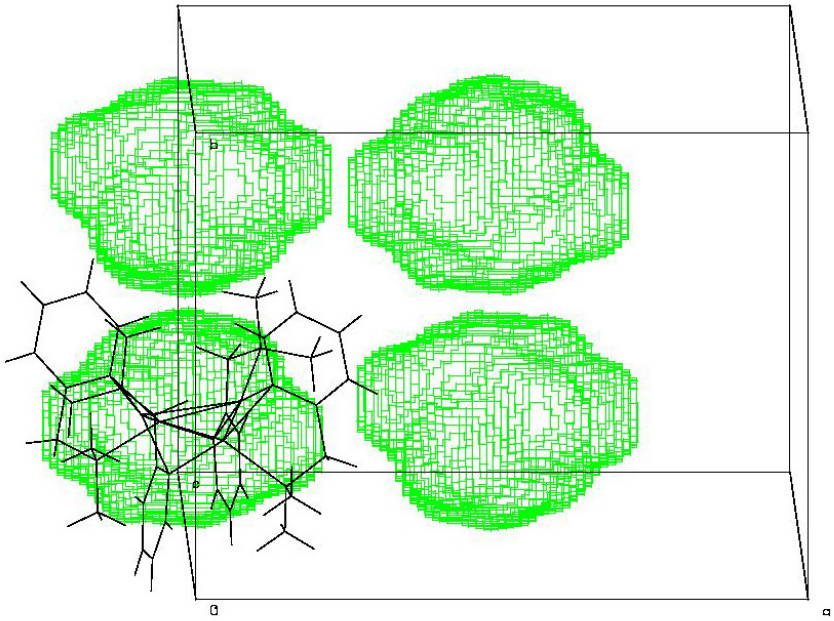


Figure 5: *Graphical display (in mono style) of the of the voids in the unitcell.*

The VOID algorithm in PLATON was also used for the description of pore types in framework structures such as Zeolites [10].

5. Alternative VOID Display

PLATON also implements an alternative way to visualize voids in a crystal structure. The version in PLATON is inspired by a program by Mugnoli [6]. With this method spheres touching the van der Waals surfaces are introduced and displayed.

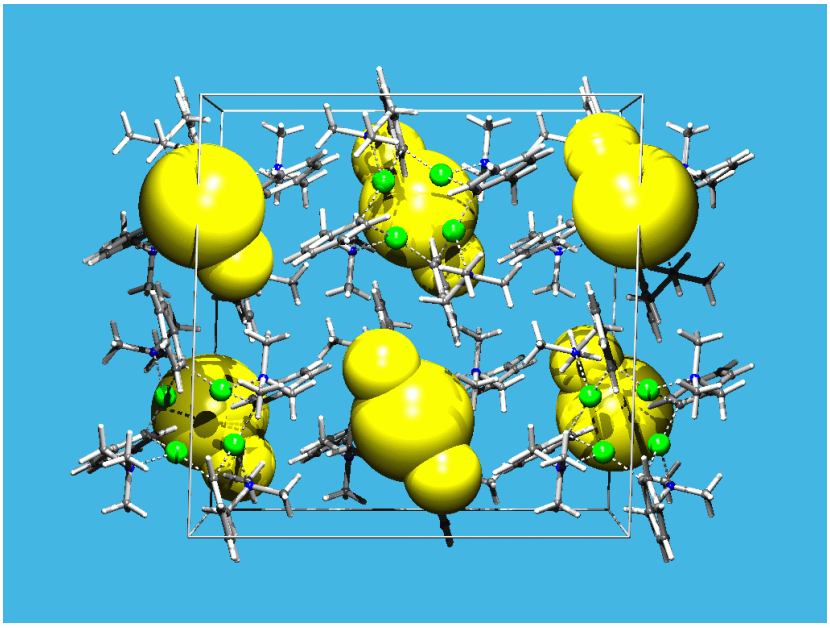


Figure 6: *Wireframe model of the ordered contents of the unitcell. The yellow spheres represent the voids in the structure. The Povray input was generated with the PLUTON link in PLATON.*

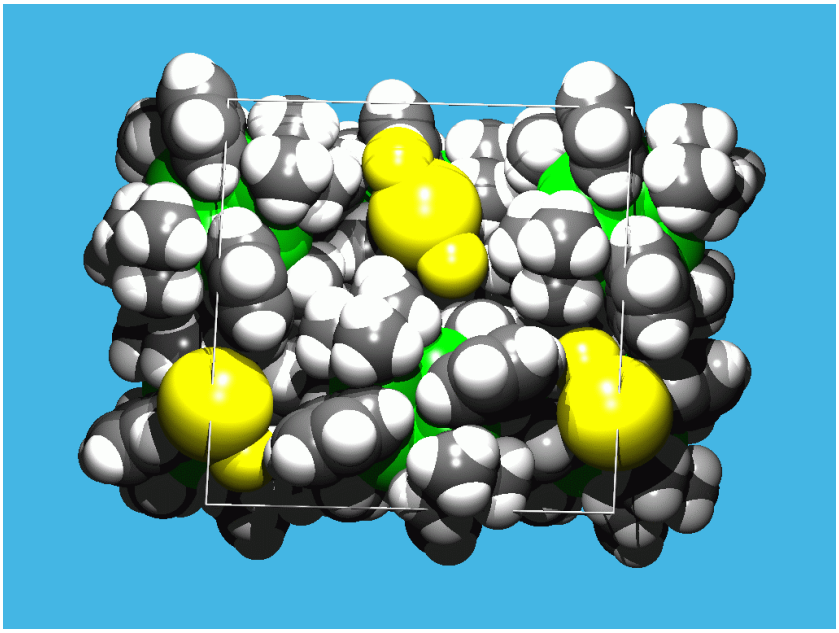


Figure 7: V*an der Waals radii presentation of the ordered part of the structure. The yellow spheres represent the voids in the structure.*

6. Concluding Remarks

Voids containing disordered solvents are often located at special positions. The solvents that occupy those sites generally have the volume needed to fill the space between the main molecules of interest but not the pointgroup symmetry compatible with the site symmetry (e.g. a THF in a 3-bar site). 'Popular' disorder sites are centres of inversion and 3, 4 and 6 fold axes.

Checking for solvent accessible voids is also done as part of the IUCr CHECKCIF structure validation procedures [2]. In practice it is found that voids and their contents are not always clear and thus can be missed. Density plateaus might evade peak search algorithms. The ultimate tool to inspect void regions is the calculation of contoured difference maps, either in terms of 2D sections or rotate able 3D maps [11].

More details can be found at http://www.cryst.chem.uu.nl/platon/

7. References

[1] P. van der Sluis & A.L.Spek (1990). Acta Cryst. C46, 883-886.
[2] A.L. Spek (2003). J. Appl. Cryst. 36, 7-13.
[3] P. van der Sluis. & A.L. Spek (1990). Acta Cryst. A46, 194-201.
[4] F.M. Richards (1985). In Methods of Enzymology, 115, 440-464.
[5] C.M.W. Ho & G.R. Marshall (1990). J. of Computer-Aided Molecular Design 4, 337-354.
[6] A. Mugnoli (1992). ECM14, Abstract 530.
[7] D.J. Watkin (1972). Acta Cryst. A28, 33-35.
[8] E. Wigderson & A.Y. Meyer (1988). Comput. Chem. 12, 237-244.
[9] A.I. Kitajgorodskij. 'Molecular Crystals and Molecules'. New York, Academic Press, 1973.
[10] H. Kuppers, F. Liebau & A.L.Spek (2006). J. Appl. Cryst. 39, 338-346.
[11] D.M. Tooke & A.L. Spek (2005) J. Appl. Cryst. 38, 572-573.

The charge flipping algorithm: a powerful and universal tool for the *a priori* solution of crystal structures in any dimension

Gervais Chapuis and Lukas Palatinus
Laboratoire de cristallographie, Ecole Polytechnique Fédérale de Lausanne, Cubotron, 1015 Lausanne, Switzerland, email: gervais.chapuis@epfl.ch and lukas.palatinus@epfl.ch WWW: http://lcr.epfl.ch/

Abstract

The charge flipping algorithm which was recently published appears to be a very powerful tool for the solution of the phase problem in crystallography. The algorithm solves the phase problem iteratively from a full set of intensities by switching between direct and Fourier space. No *a priori* information is required and only one free parameter has to be adjusted. This algorithm is remarkable owing to its universality. It has been successfully applied to solve periodic and aperiodic structures including quasicrystals and incommensurate structures, from single crystal and powder diffraction and from X-ray and neutron measurements.

Introduction

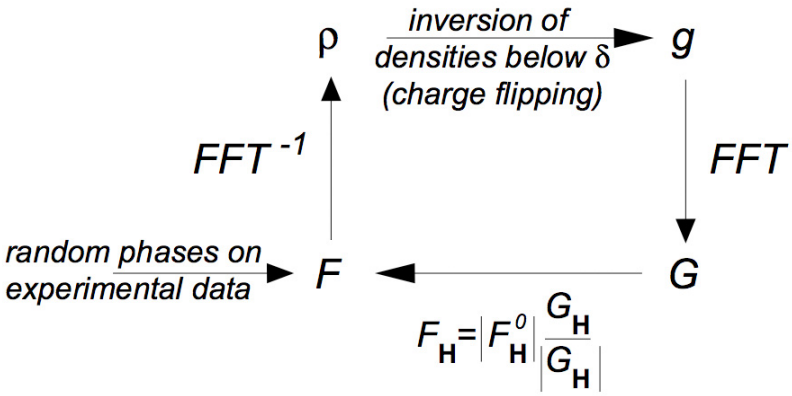
In 2004, Oszlányi and Sütő published a very interesting paper proposing a new algorithm to solve the phase problem in structural crystallography. The fact that the algorithm was very simple to describe and that it did not require any *a priori* information other than a full set of intensities was very appealing. Shortly afterwards, the same authors published an improved algorithm (2005) proposing a special treatment for the weak intensities. The simplicity of the algorithm and the ease to program it was immediately exploited and applied to aperiodic crystals (Palatinus, 2004). In this article, the author successfully applied the charge flipping algorithm to solve a number of incommensurate structures from experimental data. Later the method was also shown to work well for decagonal and icosahedral quasicrystals.

In the last decade, the search of new algorithms to solve quasi-crystalline structures has been particularly active and has generated a bright spectrum of new methods. The method proposed by Elser (1999) is looking for the set of phases that maximizes the value of the global minimum in the density. Starting from random phases, the method uses techniques of linear programming to reproduce the correct phases. The method closest to charge flipping is the low density elimination (LDE) technique. This method has been described in two successive papers by Shiono &Woolfson (1992) and Refaat & Woolfson (1993) and was initially used for removing negative peaks and sharpening peaks in the E map. This is an iterative method where positive densities are kept unchanged or slightly modified and negative densities are set to zero. In earlier publications, the method was used for phase extension and refinement of macromolecules. The method is effective when high-resolution data are available but slow in converging. Later, the method was successfully extended (see for example Matsugaki and Shiono, 2001) for the *ab initio* solution of small molecule and protein structures. Recently the method is successfully applied for the structure solution of quasicrystals (Yamamoto & Takakura, 2006).

In the next chapters of this article, we shall first present the specificities of the charge flipping (CF) algorithm and illustrate the power of the algorithm with a few examples. This algorithm is unique and distinguishes itself by its universality: it has been successful in solving any type of structures ranging from periodic to aperiodic, from powders to single crystals and from X-ray to neutron data.

The charge flipping algorithm

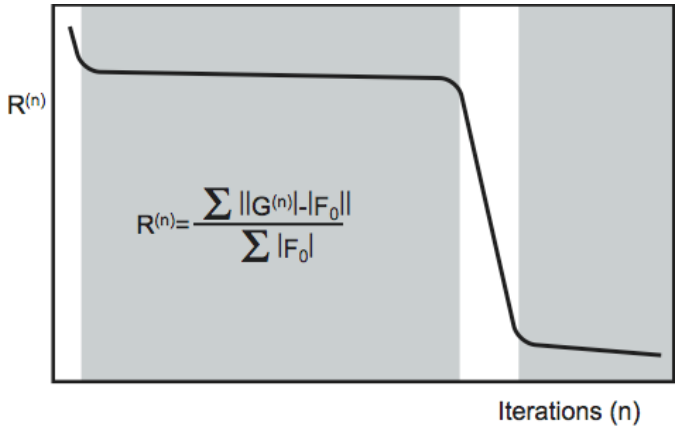
The charge flipping algorithm is an iterative process starting with a complete set of diffraction intensities with atomic resolution. The algorithm does not require any *a priori* information on the symmetry of the structure or on the chemical composition of the structure. These properties can be included in the model after the application of charge flipping and can be even to a large extent, derived directly from the result of charge flipping. The iterative process is illustrated in the following diagram:



First, a set of random phases are assigned to all the structure factors, which can be deduced, up to a scale factor, from the intensities. The inverse Fourier transform can be calculated yielding a density function. Of course, this density function ρ(**x**) exhibits positive as well as negative extrema due to the inadequate set of phases. The second step is to create a new density function g(**x**) which can be constructed from ρ(**x**) by inverting the sign of all densities falling below a certain positive threshold δ (charge flipping). The new density g(**x**) can be Fourier-transformed to obtain the complex magnitudes G(**H**) with phases φ(**H**) and amplitudes |G(**H**)|. In the last step of the cycle, new structure factors F(**H**) are obtained with the experimental magnitudes |F(**H**)| and phases φ(**H**). These structure factors are entered in the next iteration cycle.

One may wonder when the iterative process can be considered as converged. In the CF process, we distinguish in general four different phases, which are illustrated below. The first phase is shortly followed by the second phase which can be interpreted as a search in phase space. It is followed by a third phase with a sharp decrease of the residual R-value indicating that a solution of the structure has probably been found. The last phase is essentially a stabilising process.

The whole algorithm has only one free parameter, namely the flipping threshold δ. In practice, this threshold can be easily determined. Many tests on various structures show, that the process is converging rapidly within seconds or minutes on recent desktops.



One should first remark that this algorithm is dimension independent. This means that periodic as well as aperiodic structures can be solved by this process. The second remark is that the CF method is not an optimisation process in the mathematical sense, *i.e.* the algorithm is not attempting to optimise a functional of the density or phases. The third remark is that no rigorous mathematical theory is currently known that could explain the surprising efficiency of the CF algorithm.

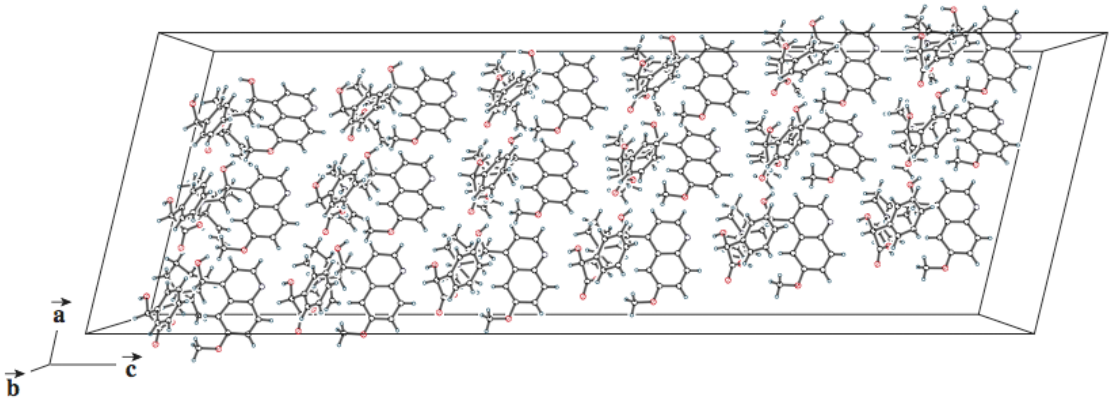
The last remark has however not refrained the interested users to apply the CF algorithm. Within a short period of time, the method has already been applied very successfully to solve a number of structures from many different fields. In the next chapter, we shall illustrate only a few examples owing to space limitation.

Examples of solutions

i. Quininium (R) mandelate (QRM)

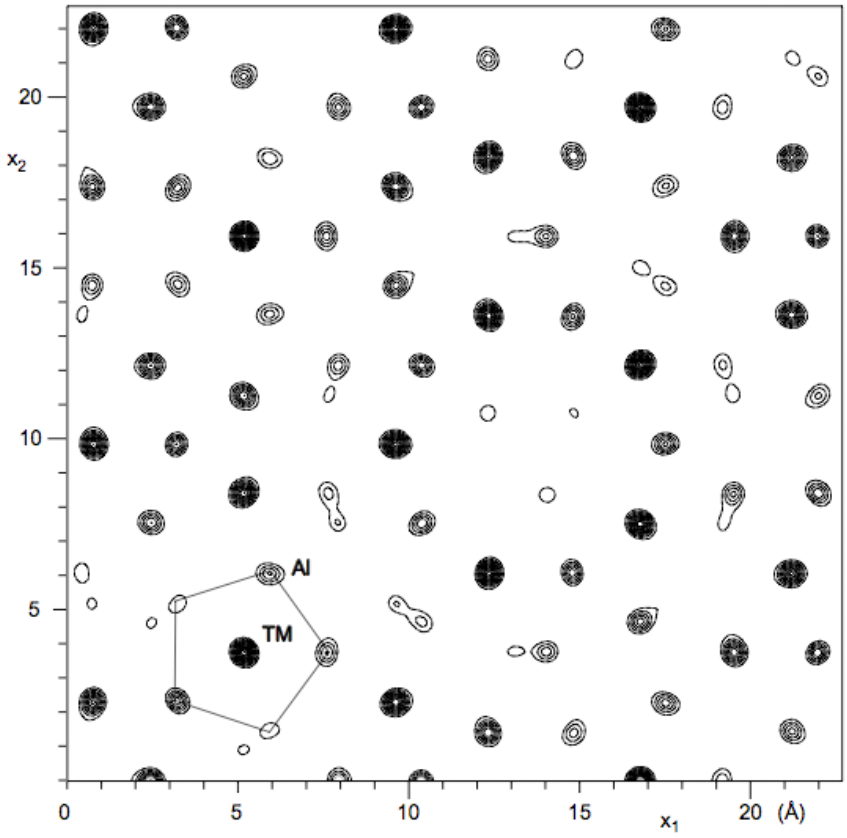
QRM is an organic incommensurate structure, with 35 non-H atoms per formula units. The representation below is a commensurate approximant with 3x6 cells. The solution of the structure (Schönleber & Chapuis, 2001) either as an approximant or in the superspace approach from single crystal diffraction data was a real challenge with the standard structure-solution methods. The displacive modulation affects essentially the quinoline moiety (two cyclohexan rings sharing a common bond) and the phenyl ring. The interested reader can observe the modulation by carefully observing the 18 QRM molecules.

The CF method could solve the structure in (3+1)d superspace practically in automatic mode.



ii. Decagonal-Quasicrystal (d-QC)

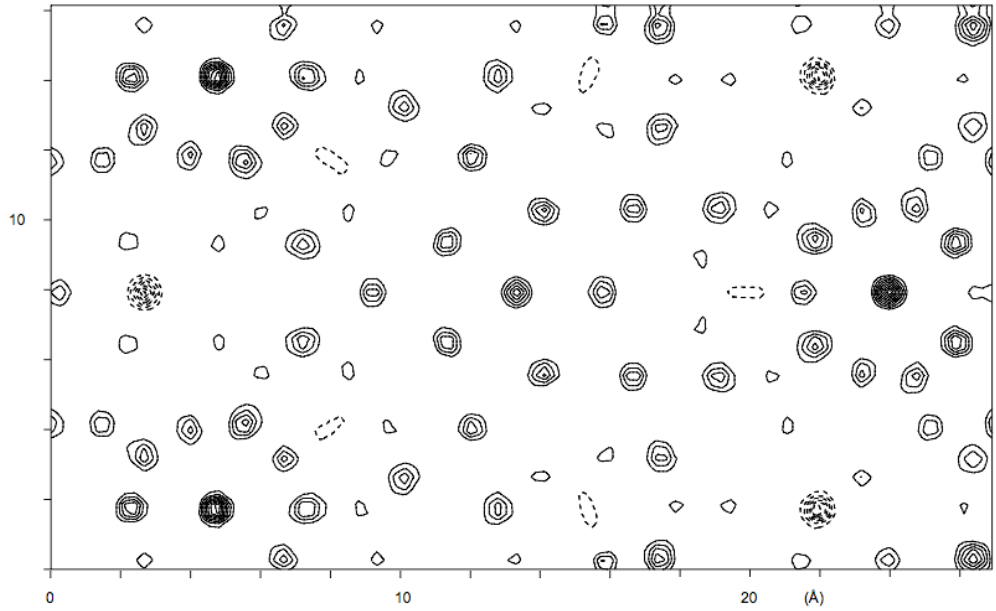
The structure of the decagonal phase Al_{70} Ni_{15} Co_{15} which has been published earlier (Steurer *et al.* 1993). The structure was solved with the software Superflip (Palatinus & Chapuis 2006) in five-dimensional space from the X-ray dataset kindly provided to us by the authors. The illustration given below represents a portion of the structure of 22.7 x 22.7 $Å^2$ which can be directly compared with fig. 7a of the original article of the authors. The x_1 -x_2 plane is perpendicular to the tenfold screw axis. The similarity between the two figures is striking and it is difficult to find any meaningful differences. Even the pairs of split atoms which are marked in the original publication can be found in the density map resulting from the charge flipping algorithm. The transition metals can be clearly distinguished from the aluminium atoms. With current desktop computers, solution convergence is reached within minutes.



The charge flipping algorithm was also successful in solving a new and unknown decagonal QC phase in the system Al-Ir-Os (Katrych et al. 2006). All the 6782 unique reflections measured experimentally were used to solve the structure on a 5- dimensional grid of 24 x 24 x 24 x 24 x 64 points. The charge flipping calculation converged after less than 100 cycles. This procedure yields clearly the heavy atoms but in order to improve the resolution of the light atoms by suppressing the noise and amplify the weak structural features, the procedure consisting of summing up many individual maps was used in the final stages of the analysis.

iii. Icosahedral-QC

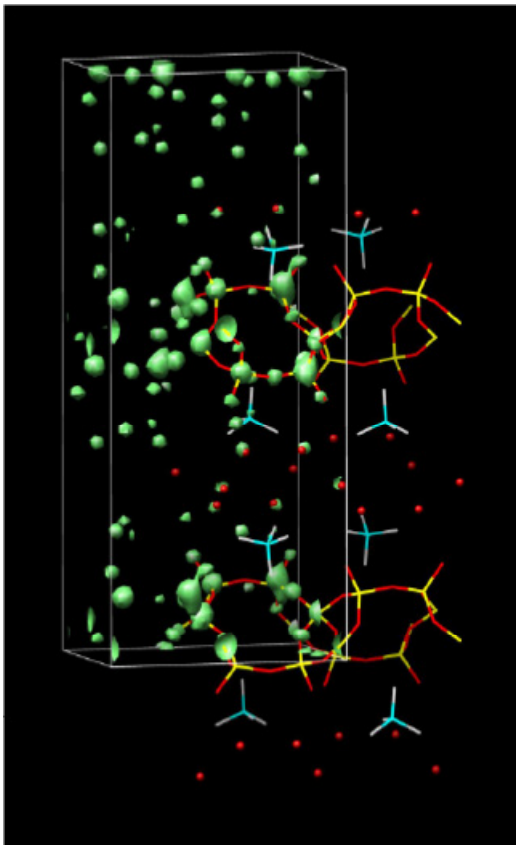
Following the above examples, it is thus clear that the charge flipping algorithm is a powerful method to solve aperiodic structures based on experimental X-ray diffraction data. What about solving any structure including aperiodic structures from neutron diffraction data? With neutron diffraction, we are facing the problem of negative scattering lengths of some atom types (H, Mn, etc.) and one may wonder if this property is compatible with the charge inversion, which is introduced during each cycle. Surprisingly, it appears that a simple modification suffices to generalize the charge flipping algorithm also for densities with negative regions. It is sufficient to modify the flipping step from flipping all density below δ to flipping the density only in the interval between $-\delta$ and $+\delta$ (Oszlányi & Sütő 2006). This is illustrated with the example of the i-$Al_{68}Pd_{23}Mn_9$ structure based on single crystal neutron diffraction data (unpublished data kindly provided by Marc de Boisieu).



The figure above represents the electron density of a portion of the structure, more specifically a slab of 2.15 Å thickness perpendicular to a five fold axis. Three types of atoms can be clearly identified. The Mn atoms exhibit a negative density whereas the Pd and Al atoms exhibit a positive density, which is proportional to their scattering lengths.

iv. Layered silicate from powder diffraction data

The structure solution from powder diffraction data faces, in addition to all problems common with the single-crystal data, the intrinsic problem of reflection overlap. As a result, not all reflection intensities are individually known and the structure solution is more complicated. Charge flipping is relatively robust to random errors in the data and thus can solve simple structures from powder diffraction data directly. However, for more complex structures an improved method is needed that would allow for repartitioning of the intensities of overlapping reflections. A method that proved particularly useful was a combination of charge flipping with histogram matching (Baerlocher *et al.* 2006). During the CF iteration the density is modified so that it matches an expected density histogram. This modified density is then used to repartition the intensities of overlapping reflections. Using this method, several complex crystal structures were solved form powder diffraction data. As an example, the figure below shows the solution of a structure of a layered silicate E-401 with 23 non-hydrogen atoms in the asymmetric unit (140 in the unit cell) with symmetry I2mb. The structure is superimposed as a stick model on the isosurface representation of the electron density. The difficulties in determining the correct symmetry hindered the solution of this structure by direct methods. Charge flipping does not need any information about the symmetry and thus the structure could be immediately solved and the correct symmetry inferred from the result.



Superflip

In order to provide a general tool for application of charge flipping algorithm for crystal-structure solution, a program named Superflip (Palatinus & Chapuis, 2006) has been developed. This program can reconstruct scattering densities from diffraction data at arbitrary dimensions, including, but not limited to, three for the standard structures, four to six for modulated structures and quasicrystals, and two, for example for the purposes of surface scattering experiments. The program allows for structure solution with x-ray and neutron scattering data both from single crystal and powder. It determines automatically the optimal value of the threshold δ, so that the user does not need to intervene and the structure solution proceeds completely automatically. After the charge flipping iteration the program can search for the position of the symmetry operations in the resulting density and average the density so that it corresponds to the symmetry expected by the user. Subsequently the resulting electron density can be analysed by the program EDMA (Smaalen *et al.* 2003) that can extract the position of the atoms from the reconstructed density and export it to the formats of Jana2000, SHELX, or to CIF. Both Superflip and EDMA can be downloaded free of charge at http:/superspace.epfl.ch/superflip/.

Thanks to the simplicity and usefulness of the algorithm several other implementations have been created. The most notable one is probably the "Flipper" option in the Platon package, but many other programs that are used mainly by their authors only have been written, too. We have not extensively tested any of these programs and thus we do not feel competent to comment on them.

Conclusion

Since the discovery of X-ray diffraction, the solution of the phase problem has forced generations of specialists to develop new methods to circumvent the shortcomings of diffraction, *i.e.* the loss of the phases of the structure factors in diffraction experiments. The Patterson method followed by direct methods and all its variants both contributed to sudden increases in determining new structures. We hope that the charge flipping method and its improved variants will also contribute to the next impulse in this field. The important point here is that contrary to previous methods, the CF method and the related

density-modification methods are *a priori* methods which do not require any assumptions on the atomicity of the structure or the knowledge of its symmetry and chemical composition. The unique property of the CF algorithm is its ability to solve within the same formalism not only periodic but also aperiodic structures. Although our experience is still limited, all the attempts to solve incommensurate structures, dodecahedral or icosahedral quasicrystals with experimental data were successful. Neutron diffraction data with negatively scattering elements is not a limitation either. We have also seen some successful examples of CF solutions from powder diffraction experiments. The next challenge of the method will then be the solution of the diffraction from single objects.

References

Baerlocher Ch., McCusker, L. B. and Palatinus, L. "Charge flipping combined with histogram matching to solve complex crystal structures from powder diffraction data", Zeitschrift für Kristallographie, in press (2006)

Elser, V. "X-ray phase determination by the principle of minimum charge." Acta Crystallographica Section A 55: pp. 489-499 (1999).

Katrych, S., Weber, T., Kobas, M., Massüger, L., Palatinus, L., Chapuis, G. and Steurer, W. "New stable decagonal quaiscrystal in the system Al-Ir-Os. To be published in "Journal of Alloys and Compounds" (2006).

Matsugaki, N. and Shiono, M. "Ab initio structure determinations by direct-space methods: tests of low-density elimination." Acta Crystallographica Section D 57: pp. 95-100 (2001).

Oszlányi, G. and Sütő, A. "Ab initio structure solution by charge flipping." Acta Crystallographica Section A 60: pp. 134-141 (2004).

Oszlányi, G. and Sütő, A. "Ab initio structure solution by charge flipping. II. Use of weak reflections." Acta Crystallographica Section A 61: pp. 147-152 (2005).

Oszlányi, G. and Sütő, A. "How can we cope with negative scattering density?" ECM23 Leuven, Book of Abstracts (2006).

Palatinus, L. "Ab initio determination of incommensurately modulated structures by charge flipping in superspace." Acta Crystallographica Section A 60: pp. 604-610 (2004).

Palatinus, L. and Chapuis, G. "Superflip - a computer program for solution of crystal structures by charge flipping in arbitrary dimensions", http:/superspace.epfl.ch/superflip (2006)

Refaat, L. S. and Woolfson, M. M. "Direct-space methods in phase extension and phase determination. II. Developments of low-density elimination." Acta Crystallographica Section D 49: pp. 367-371 (1993).

Schönleber, A. and Chapuis, G.. "Quininium (R)-mandelate, a structure with large Z' described as an incommensurately modulated structure in (3+1)-dimensional superspace." Acta Crystallographica Section B 60(1): pp. (2004).

Shiono, M. and Woolfson, M. M. "Direct-space methods in phase extension and phase determination. I. Low-density elimination." Acta Crystallographica Section A 48: pp. 451-456 (1992).

Steurer, W., Haibach, T., Zhang, B., Kek, S. and Luck, R. "The structure of decagonal $Al_{70}Ni_{15}Co_{15}$." Acta Crystallographica Section B 49: pp. 661-675 (1993).

van Smaalen, S., Palatinus, L., and Schneider, M. "The Maximum Entropy Method in superspace", Acta Crystallographica Section A 59: pp. 459-469 (2003)

Yamamoto, A. and Takakura, H. "Recent development in structure determinations for quasicrystals." Philosophical Magazine 86(3-5): pp. 405-411 (2006).

cctbx news

Ralf W. Grosse-Kunstleve[a], Peter H. Zwart[a], Pavel V. Afonine[a], Thomas R. Ioerger[b], Paul D. Adams[a]

[a]Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA and [b]Texas A&M University, College Station, TX 77843, USA. E-mail: rwgk@cci.lbl.gov ; WWW: http://cci.lbl.gov/

Abstract

We describe recent developments of the Computational Crystallography Toolbox.

Preamble

In order to interactively run the examples scripts shown below, the reader is highly encouraged to visit http://cci.lbl.gov/cctbx_build/ and to download one of the completely self-contained, self-extracting binary cctbx distributions (supported platforms include Linux, Mac OS X, Windows, IRIX, and Tru64 Unix). All example scripts shown below were tested with cctbx build 2006_11_22_0037.

In the following we refer to our articles in the previous editions of this newsletter as "Newsletter No. 1", "Newsletter No. 2", etc. to improve readability. The full citations are included in the reference section.

1 Introduction

The *Computational Crystallography Toolbox* (cctbx, http://cctbx.sourceforge.net/) is the open-source component of the Phenix project (http://www.phenix-online.org/). Most recent cctbx developments are geared towards supporting new features of the `phenix.refine` application. Thus, the open-source `mmtbx` (macromolecular toolbox) module is currently being most rapidly developed. In this article we give an overview of some of the recent developments. However, the main theme of this article is the presentation of a light-weight example command-line application that was specifically developed for this newsletter: sequence alignment and superposition of two molecules read from files in PDB format. This involves parameter input based on the Phil module presented in Newsletter No. 5, fast reading of the PDB files with the new `iotbx.pdb.input` class, simple sequence alignment using the new `mmtbx.alignment` module, and use of the Kearsley (1989) superposition algorithm to find the least-squares solution for superposing C-alpha positions. The major steps are introduced individually, followed by a presentation of the complete application.

The example application is deliberately limited in functionality to make it concise enough for this article. The main goal is to show how the open-source components are typically combined into an application. Even though the example is quite specific to macromolecular crystallography, we believe it will also be useful for a small-molecule audience interested in utilizing the large open-source library of general crystallographic algorithms (see our previous articles in this newsletter series) to build an application.

2 iotbx.pdb

The PDB format is the predominant working format for atomic parameters (coordinates, occupancies, displacement parameters, etc.) in macromolecular crystallography, but many small-molecule programs also support this format. `phenix.refine` also utilized the PDB format, mainly to facilitate easy communication with other programs, most notably graphics programs for visualization.

2.1 iotbx.pdb.input

The PDB format specifications are available at http://www.pdb.org/ . Technically, the format is very simple, therefore a vast number of parsers exist in scientific packages. The cctbx is no exception. A parser implemented in Python has been available for several years. In many cases Python's runtime performance is sufficient for interactive processing of PDB files, but can be limiting for large files, or for traversing the entire PDB database (currently 40731 files, about 25 GB total). This has prompted us to implement a fast parser in C++, complete with Python bindings in the same style as all other cctbx C++ classes, comprehensive error reporting, and fully automatic memory life-time management (no manual new/delete or malloc/free). Reading a PDB file from Python is simple:

```
import iotbx.pdb
pdb_inp = iotbx.pdb.input(file_name="pdb1htq.ent")
```

With a size of 76 MB this is the largest file in the PDB, but full processing takes only 3.8 s on a 2.6 GHz Opteron, of which about 0.5 s are for simply transferring the data from disk into memory. In contrast, the older Python implementation needs 89 s for processing the file into data structures of similar complexity. In the future all our cctbx-based applications will make use of the new, faster parser. Interestingly, when processing the PDB database with `iotbx.pdb.input` using multiple CPUs, disk-I/O is the rate limiting step. Using 8 CPUs, we can process all 25 GB in less than five minutes. Using more CPUs does not reduce this time.

The `pdb_inp` object holds the information from the PDB file in a structured way. The "sections" of the file according to the PDB format specifications at http://www.pdb.org/ are available as, e.g.:

```
pdb_inp.title_section()
pdb_inp.remark_section()
pdb_inp.crystallographic_section()
...
```

In Python these sections appear as simple lists of strings. The full power of Python and the cctbx libraries is available for post-processing this information. Since most sections are never very large, there is no point in writing specialized C++ processing code, which is typically significantly more labor intensive compared to writing equivalent Python code, and much more difficult to adjust for new developments.

The only section of PDB files that is sometimes found to be very large is the "coordinate section" with the ATOM and HETATM records. This section is fully processed in the `iotbx.pdb.input` constructor shown above. The corresponding information is available via the methods:

```
labels_list = pdb_inp.input_atom_labels_list()
atoms = pdb_inp.atoms()
```

which return arrays of the same length, each with one data object per atom. The information for one atom is again accessible from Python, e.g.:

```
for labels,atom in zip(labels_list, atoms):
  print labels.chain(), labels.resname(), labels.name(), atom.xyz, atom.b
```

2.2 iotbx.pdb.hierarchy

The `input_atom_labels` objects in the `pdb_inp.input_atom_labels_list()` above store the `name`, `resname`, `chain`, `icode` (insertion code), `segid` (segment identifier), and `altloc` (alternate location indicator) for each atom. This information defines a hierarchical organization of the macromolecules, but in a highly redundant way which complicates further processing steps, such as the assignment of geometry restraints (see Newsletter No. 4). `iotbx.pdb.input` supports building an `iotbx.pdb.hierarchy` object. The redundant atom labels are analyzed to build a non-redundant six-deep `hierarchy` object, e.g.:

```
hierarchy = pdb_inp.construct_hierarchy()
```

The six-deep data structure consists of:

```
hierarchy
  model
    chain
      conformer
        residue
          atom
```

which can be concisely traversed from Python:

```
for model in hierarchy.models():
  for chain in model.chains():
    for conformer in chain.conformers():
      for residue in conformer.residues():
        for atom in residue.atoms():
          # ...
```

The time for building the `hierarchy` object given `pdb1htq.ent` is about 0.7 s. Traversing the hierarchy with the five-deep loop above takes only about 0.6 s, i.e. is unlikely to be a rate-limiting step even for very large structures. Therefore the few lines of example code given in this section are probably one of the most convenient and efficent ways of quickly processing a PDB file from a scripting language.

For general information on how to learn more about Python objects, look under the "Tutorials Siena 2005" link at cctbx.sf.net. For example, the command:

```
libtbx.help iotbx.pdb.residue
```

will show the complete interface of the `residue` object.

`hierarchy` objects can be manipulated or constructed from scratch from both Python and C++. However, high-level functionality like inserting or deleting residues or chains, or formatting output is currently not available. We will add such high-level manipulations as the need arises. The typical development process is to implement a required high-level operation given the currently available interfaces, then add it as a new method to the most suitable existing class to make it easily accessible for other purposes. We expect the hierarchy objects to continuously grow in this way for some time to come.

2.3 pdb_inp.xray_structure_simple

The `xray_structure_simple` simple method of `iotbx.pdb.input` is an efficient implementation converting the information stored in the `pdb_inp` object above to a list of `cctbx.xray.scatterer` objects, managed by the `cctbx.xray.structure` class. See our previous newsletter articles and the Siena 2005 tutorials for various examples on how to work with these objects.

Fundamentally, the conversion is trivial. Each input atom is converted to exactly one `xray.scatterer`. However, as always, the devil is in the details. The PDB `CRYST1` and `SCALE` cards have to be evaluated to obtain the correct fractionalization matrix (PDB coordinates are with respect to a Cartesian basis). The trickest problem is the determination of the scattering type for each atom, for which three PDB columns have to be considered (atom name, element symbol, charge). Following the PDB format specifications strictly, the scattering type is clearly defined, but unfortunately deviations from the strict specifications are quite common. For example, the element symbol may be missing or mis-aligned, or the charge symbol is sometimes found to be given as "+2" instead of "2+". The `xray_structure_simple` method allows for some deviations from the strict PDB specifications as long as the error is highly obvious. More serious errors are communicated via exceptions with carefully formatted, informative error messages.

The `xray_structure_simple` method is relatively expensive in terms of runtime, partially because the site symmetry is determined for each atom, which involves looping over the symmetry elements and distance calculations. The runtime for the `pdb1htq.ent` structure (978720 atoms) is about 9.9 s. However, this step is typically performed only once at the start of a program. To put this further into context, a structure factor calculation up to a resolution of 3 A, using the FFT method, takes about 26 s. This underlines that the time for I/O using the new `pdb.input` class is generally neglible in the context of a whole application.

For completeness, the code for building the `xray.structure` and computing the structure factors is:

```
xray_structure = pdb_inp.xray_structure_simple()
xray_structure.structure_factors(d_min=3, algorithm="fft")
```

3 mmtbx.alignment

`mmtbx.alignment` provides algorithms for aligning two protein sequences, where each sequence is represented as a string of one-letter amino-acid codes. The implementation is based on the ideas of Gotoh (1982) and runs in quadratic time O(M*N), where M and N are the sequence lengths. It does both global (Needleman & Wunsch, 1970) and local (Smith & Waterman, 1981) alignments, assuming affine (linear) gap penalties (for which default gap-cost parameters may be changed by the user). Alignments are based on maximizing similarity. Similarity scores between amino acids are specified via symmetric matrices. Similarity matrices of Dayhoff (1978) and BLOSUM50 (Henikoff & Henikoff (1992), http://en.wikipedia.org/wiki/BLOSUM) are provided. User-supplied matrices are also supported (this feature also enables alignment of non-amino-acid sequences).

To show a short example of aligning two sequences:

```
from mmtbx.alignment import align
align_obj = align(
  seq_a="AESSADKFKRQHMDTEGPSKSSPTYCNQMM",
  seq_b="DNSRYTHFLTQHYDAKPQGRDDRYCESIMR")
```

The `align_obj` holds matrices used in the dynamic-programming (http://en.wikipedia.org/wiki/Dynamic_programming) alignment algorithm. Methods are available to get the alignment score and to extract the actual sequence alignment:

```
print "score: %.1f" % align_obj.score()

alignment = align_obj.extract_alignment()
print alignment.match_codes
print alignment.a
print alignment.matches(is_similar_threshold=0)
print alignment.b
```

The output is:

```
score: 6.0
mmmmmmmmmmmmmmmmmmmmmmmmmmmmmmmm
AESSADKFKRQHMDTEGPSKSSPTYCNQMM
  |    |  || |               |
DNSRYTHFLTQHYDAKPQGRDDRYCESIMR
```

`match_codes` is a string of the characters `m`, `i`, and `d`, for match, insertion, and deletion, respectively. The alignment above is based on identity matches only. Alternatively, the similarity matrices of `"dayhoff"` or `"blosum50"` can be used, e.g.:

```
align_obj = align(
  seq_a="AESSADKFKRQHMDTEGPSKSSPTYCNQMM",
  seq_b="DNSRYTHFLTQHYDAKPQGRDDRYCESIMR",
```

```
  style="global",
  gap_opening_penalty=10,
  gap_extension_penalty=2,
  similarity_function="blosum50")
```

The output of the same print statements as above is in this case:

```
mmmmmmmmmmmmmmmmmdmmmmmmmmmmmmmmi
AESSADKFKRQHMDTEGPSKSSPTYCNQMM-
  |    |  || | * |      ||  *|
DNSRYTHFLTQHYDAK-PQGRDDRYCESIMR
```

Here the `*` indicate residues that are similar above the `is_similar_threshold`, using the `blosum50` similarity matrix.

Alignment of sequences of typical lengths is fast enough for interactive work (300 residues: about 1 s), but can take minutes given thousands of residues (2200 residues: about 40 s). Also, for very long sequences the memory consumption can be significant since four M*N alignment matrices are stored as pure Python objects. In the future we may reimplement the core of the alignment algorithm in C++ to increase runtime performance (by a factor 30-50) and to significantly reduce memory consumption. However, the Python interface presented here is expected to stay the same.

4 scitbx.math.superpose

The `scitbx.math.superpose` module implements the quaternion method of Kearsley (1989) for superpositioning two related vector sets. In comparison to the related method of Kabsch (1976), this method has the advantage of gracefully handling degenerate situations (e.g. if all atoms are on a straight line) without the need for handling special cases (Kabsch, 1978). For all non-degenerate situations, the results of the Kabsch and Kearsley methods are identical within floating point precision.

We will give a self-contained example, using the `iotbx.pdb.input` class discussed above. First, we define a few atoms to be aligned:

```
  # residues in PDB entry 1AON
  gly1 = """\
ATOM     55  N   GLY A   9      47.072 -70.250  -4.389  1.00 27.28
ATOM     56  CA  GLY A   9      45.971 -69.823  -3.545  1.00 22.53
ATOM     57  C   GLY A   9      45.946 -68.397  -3.056  1.00 25.45
ATOM     58  O   GLY A   9      46.350 -67.467  -3.764  1.00 28.17
""".splitlines()
  gly2 = """\
ATOM    134  N   GLY A  19      46.795 -55.602  -6.961  1.00 15.66
ATOM    135  CA  GLY A  19      47.081 -55.164  -8.320  1.00 11.86
ATOM    136  C   GLY A  19      45.844 -54.551  -8.936  1.00  7.75
ATOM    137  O   GLY A  19      45.851 -53.398  -9.384  1.00 10.45
""".splitlines()
```

This code produces two Python lists of Python strings. To be compatible with the `iotbx.pdb.input` constructor, these have to be converted to C++ arrays ("flex arrays", see Newsletter No. 1):

```
from cctbx.array_family import flex
gly1 = flex.std_string(gly1)
gly2 = flex.std_string(gly2)
```

Now we are ready to instantiate two `iotbx.pdb.input` objects. Instead of reading directly from a file as shown before, we read the PDB `lines` from the C++ array of strings:

```
import iotbx.pdb
pdb1 = iotbx.pdb.input(source_info=None, lines=gly1)
pdb2 = iotbx.pdb.input(source_info=None, lines=gly2)
```

Next, we extract two C++ flex arrays with the coordinates:

```
sites1 = pdb1.extract_atom_xyz()
sites2 = pdb2.extract_atom_xyz()
```

These arrays can be used directly to instantiate the `least_squares_fit` class which computes the rotation and translation for the best fit using the algorithm of Kearsley (1989) (which is the default):

```
from scitbx.math import superpose
superposition = superpose.least_squares_fit(
  reference_sites=sites1,
  other_sites=sites2)
```

The `r` and `t` attributes are `scitbx.matrix` instances which provide `mathematica_form()` methods which we use here for pretty-printing:

```
print superposition.r.mathematica_form(
  label="r", one_row_per_line=True, format="%8.5f")
print superposition.t.mathematica_form(
  label="t", format="%8.5f")
```

The output is:

```
r={{-0.72436, -0.03369,  0.68860},
   {-0.43741,  0.79448, -0.42127},
   {-0.53289, -0.60635, -0.59023}}
t={{83.88229}, {-8.78874}, {-17.07881}}
```

To compute the RMS difference of the superposed sites:

```
sites2_fit = superposition.other_sites_best_fit()
print "rms difference: %.4f" % sites1.rms_difference(sites2_fit)
```

Output:

```
rms difference: 0.3671
```

The following code produces a listing of distances for each atom:

```
from scitbx import matrix
lbls1 = pdb1.input_atom_labels_list()
lbls2 = pdb2.input_atom_labels_list()
for l1,s1,l2,s2f,d in zip(lbls1, sites1,
                          lbls2, sites2_fit,
                          sites2_fit-sites1):
  print l1.pdb_format(), "%8.4f, %8.4f, %8.4f" % s1
  print l2.pdb_format(), "%8.4f, %8.4f, %8.4f" % s2f
  print "      difference: %8.4f, %8.4f, %8.4f" % d, \
        "|d| = %6.4f" % abs(matrix.col(d))
  print
```

The input atom labels are obtained from the `iotbx.pdb.input` objects using the `pdb_format()` method which returns the atom labels in the same arrangement as found in the PDB file; this output is useful for locating an atom in the original file. `sites2_fit-sites1` uses flex array algebra to compute the difference vectors, and the `scitbx.matrix` class is used to compute the length of the vector. The output starts with:

```
" N   GLY A   9 "  47.0720, -70.2500,  -4.3890
" N   GLY A  19 "  47.0655, -70.5000,  -4.1925
      difference:  -0.0065,  -0.2500,   0.1965 |d| = 0.3180
...
```

It is easy to verify that the Kabsch method gives identical results for this non-degenerate configuration of atoms:

```
superposition_kabsch = superpose.least_squares_fit(
  reference_sites=sites1,
  other_sites=sites2,
  method="kabsch")
```

In this case the rotation matrix and the translation vector are exactly identical to the Kearsley results shown above. However, `method="kabsch"` should not be used in applications since we didn't spent the effort of writing code for special cases. Therefore the Kearsly method is the default.

5 Putting the pieces together: mmtbx.super

The large variety of tools in the `cctbx` enables quick develop of small scripts (a.k.a. jiffies) that perform non-trivial tasks with relative ease.

In a typical jiffy, most work goes into building a user interface that facilitates the communication between program and user. The actual computational work that needs to be done is often not very laborious, and can be as trivial as a simple call of a library function.

The following paragraphs will illustrate the rapid development of a lightweight structure superposition command line tool using a variety of recent additions to the cctbx. The full code can be found in `cctbx_sources/mmtbx/mmtbx/command_line/super.py`. This script is also available from the command line under the name `mmtbx.super`.

5.1 Design

The goal is to develop a simple tool that carries out the following tasks:

1. Determine input file names and alignment parameters (user interface).
2. Read in two related PDB files.
3. Determine corresponding residues between the two PDB files.
4. Compute a least-squares superposition of C-alpha atoms.
5. Write out the superposed coordinates to a new PDB file.

5.2 User interface

A basic, yet versatile, user interface can be implemented in a very straightforward manner using Phil. Since Phil has been discussed at length in Newsletter No. 5, we limit ourselves to a brief overview of the implementation.

The Phil "master parameters" definition embedded in `super.py` is:

```
import libtbx.phil
master_params = libtbx.phil.parse("""\
super {
  fixed = None
    .type = str
  moving = None
    .type = str
  moved = "moved.pdb"
    .type = str
  alignment_style = *local global
    .type = choice
```

```
  gap_opening_penalty = 20
    .type = float
  gap_extension_penalty = 2
    .type = float
  similarity_matrix = *blosum50 dayhoff
    .type = choice
}
""")
```

`master_params` is a Phil `scope` instance with a `super` sub-scope (for clarity) that defines the parameters we need. `fixed` and `moving` are input files names, `moved` is an output file name. The other parameters are for `mmtbx.alignment.align`.

All parameters can be modified from the command line, e.g.:

```
mmtbx.super fixed=first.pdb moving=second.pdb similarity_matrix=dayhoff
```

This is enabled with the following code fragments in `super.py`:

```
import libtbx.phil.command_line
phil_objects = []
argument_interpreter = libtbx.phil.command_line.argument_interpreter(
  master_params=master_params, home_scope="super")

...

    try: command_line_params = argument_interpreter.process(arg=arg)
    except: raise Sorry("Unknown file or keyword: %s" % arg)
    else: phil_objects.append(command_line_params)
```

As an added convenience, bare file names are also recognized, e.g. this is an alternative to the command above:

```
mmtbx.super first.pdb second.pdb similarity_matrix=dayhoff
```

The complete code (including the fragments above) for supporting this generality is:

```
def run(args):
  phil_objects = []
  argument_interpreter = libtbx.phil.command_line.argument_interpreter(
    master_params=master_params, home_scope="super")
  fixed_pdb_file_name = None
  moving_pdb_file_name = None
  for arg in args:
    if (os.path.isfile(arg)):
      if (fixed_pdb_file_name is None): fixed_pdb_file_name = arg
      elif (moving_pdb_file_name is None): moving_pdb_file_name = arg
      else: raise Sorry("Too many file names.")
    else:
      try: command_line_params = argument_interpreter.process(arg=arg)
      except: raise Sorry("Unknown file or keyword: %s" % arg)
      else: phil_objects.append(command_line_params)
```

At this point we have to consolidate the two possible sources of information: bare file names (stored under `fixed_pdb_file_name` and `moving_pdb_file_name`) and assignments via `fixed=...` or `moving=...`. First, we combine all the Phil assignments (stored under `phil_objects`) into one `working_params` object and use the `extract()` method to get easy access to the definitions (see Newsletter No. 5):

```
working_params = master_params.fetch(sources=phil_objects)
params = working_params.extract()
```

Now we override the Phil assignments with the bare file names if available, or generate an error message if a file name is missing:

```
if (fixed_pdb_file_name is None):
  if (params.super.fixed is None): raise_missing("fixed")
else:
  params.super.fixed = fixed_pdb_file_name
if (moving_pdb_file_name is None):
  if (params.super.moving is None): raise_missing("moving")
else:
  params.super.moving = moving_pdb_file_name
```

`raise_missing()` is a simple function raising an informative exception (see `super.py`), e.g.:

```
Sorry: Missing file name for moving structure:
  Please add
    moving=file_name
  to the command line to specify the moving structure.
```

5.3 Processing of PDB input files

With all the input parameters consolidated in the `params` object above, reading in the PDB files is simple:

```
fixed_pdb = iotbx.pdb.input(file_name=params.super.fixed)
moving_pdb = iotbx.pdb.input(file_name=params.super.moving)
```

For both files we have to extract the sequence of residue names and corresponding C-alpha coordinates. This is implemented as a function that we call twice:

```
fixed_seq, fixed_sites, fixed_site_flags = extract_sequence_and_sites(
  pdb_input=fixed_pdb)
moving_seq, moving_sites, moving_site_flags = extract_sequence_and_sites(
  pdb_input=moving_pdb)
```

For the complete implementation of `extract_sequence_and_sites()` please refer to `super.py`. For simplicity, the function only considers the first `MODEL` in the pdb file, and for each chain only the first conformer (as derived from the `altloc` symbols):

```
model = pdb_input.construct_hierarchy().models()[0]
for chain in model.chains():
  selected_residues = chain.conformers()[0].residue_class_selection(
    class_name="common_amino_acid")
  residues = chain.conformers()[0].residues()
  for ires in selected_residues:
    ...
```

Another simplification is the selection of `"common_amino_acid"` residues only. For these, the residue names are translated to one-letter codes which are collected in a `seq` list:

```
import mmtbx.amino_acid_codes
...
  seq = []
  ...
      resi = residues[ires]
      resn = resi.name[0:3]
      single = mmtbx.amino_acid_codes.one_letter_given_three_letter[resn]
      seq.append(single)
```

The rest of the body of the loop over the selected residues extracts the C-alpha coordinates if available:

```
from cctbx.array_family import flex
...
  sites = flex.vec3_double()
  use_sites = flex.bool()
  ...
      use = False
      xyz = (0,0,0)
      for atom in resi.atoms():
        if (atom.name == " CA "):
          xyz = atom.xyz
          use = True
          break
      sites.append(xyz)
      use_sites.append(use)
```

The coordinates are stored under `sites`. A corresponding `use_sites` array of bools (False or True) stores if a C-alpha atom was found or not. Finally the collected sequence, coordinates and use flags are returned with:

```
return "".join(seq), sites, use_sites
```

The list of one-letter codes is converted to a plain string on the fly. The plain string is more convenient to work with in the following steps.

5.4 Sequence alignment

Sequence alignment is now a simple call of `mmtbx.alignment.align` as discussed before. The function call parameters are taken directly from the Phil `params` object:

```
align_obj = mmtbx.alignment.align(
  seq_a=fixed_seq,
  seq_b=moving_seq,
  gap_opening_penalty=params.super.gap_opening_penalty,
  gap_extension_penalty=params.super.gap_extension_penalty,
  similarity_function=params.super.similarity_matrix,
  style=params.super.alignment_style)
```

From the `align_obj` we extract the `alignment` as shown before, but we spend a little more effort to produce nice output:

```
alignment = align_obj.extract_alignment()
matches = alignment.matches()
equal = matches.count("|")
similar = matches.count("*")
total = len(alignment.a) - alignment.a.count("-")
alignment.pretty_print(
  matches=matches,
  block_size=50,
  n_block=1,
  top_name="fixed",
  bottom_name="moving",
  comment="""... see super.py ... """)
```

This code produces, e.g.:

```
The alignment used in the superposition is shown below.

The sequence identity (fraction of | symbols) is 55.1%
of the aligned length of the fixed molecule sequence.
```

```
The sequence similarity (fraction of | and * symbols) is 75.5%
of the aligned length of the fixed molecule sequence.

              12345678901234567890123456789012345678901234567890

fixed         VTDNIMKHSKNPIIIVVSNPLDIMTHVAWVRSGLPKERVIGMAGVLDAA
              *  ||*||| * ||*|||||*|**|*|||  ||||  |*||    ||*|
moving        IIPNIVKHSPDCIILVVSNPVDVLTYVAWKLSGLPMHRIIGSGCNLDSA
```

5.5 Least-squares superposition

To keep this example simple, in the least-squares superposition we want to use only the C-alpha coordinates of matching residues, i.e. residues with a `|` or `*` symbol in the output above. This information is stored under `matches` as obtained above. We also have to check if the C-alpha coordinates are available for both of the matching residues. This information is stored under `fixed_site_flags` and `moving_site_flags`. The matching + available C-alpha coordinates are obtained with this code:

```
fixed_sites_sel = flex.vec3_double()
moving_sites_sel = flex.vec3_double()
for ia,ib,m in zip(alignment.i_seqs_a, alignment.i_seqs_b, matches):
  if (m not in ["|", "*"]): continue
  if (fixed_site_flags[ia] and moving_site_flags[ib]):
    fixed_sites_sel.append(fixed_sites[ia])
    moving_sites_sel.append(moving_sites[ib])
```

Computing the superposition and printing out the RMSD between the aligned, superposed C-alpha atoms is now very simple:

```
lsq_fit = superpose.least_squares_fit(
  reference_sites=fixed_sites_sel,
  other_sites=moving_sites_sel)
rmsd = fixed_sites_sel.rms_difference(lsq_fit.other_sites_best_fit())
print "  RSMD between the aligned C-alpha atoms: %.3f" % rmsd
```

5.6 Export of moved coordinates

As the final step, `mmtbx.super` applies the rotation and translation obtained in the least squares fit to all original coordinates in `moving_pdb` and writes out the modified atom records:

```
print "Writing moved pdb to file: %s" % params.super.moved
out = open(params.super.moved, "w")
for serial, label, atom in zip(moving_pdb.atom_serial_number_strings(),
                               moving_pdb.input_atom_labels_list(),
                               moving_pdb.atoms()):
  print >> out, iotbx.pdb.format_atom_record(
    record_name={False: "ATOM", True: "HETATM"}[atom.hetero],
    serial=int(serial),
    name=label.name(),
    altLoc=label.altloc(),
    resName=label.resname(),
    resSeq=label.resseq,
    chainID=label.chain(),
    iCode=label.icode(),
    site=lsq_fit.r * matrix.col(atom.xyz) + lsq_fit.t,
    occupancy=atom.occ,
    tempFactor=atom.b,
    segID=atom.segid,
    element=atom.element,
    charge=atom.charge)
```

All of the information on the input ATOM or HETATM records is passed through as-is, except for the coordinates. We make use of the `scitbx.matrix` facilities again to apply `lsq_fit.r` and `lsq_fit.t` to

`atom.xyz`. `format_atom_record()` is a very simple function, essentially just a Python string formatting statement which could also be spelled out inline. However, the assignment of the data items to function parameters is easier to read and understand than the raw formatting statement, and the function handles some subtleties (e.g. overflowing serial and resSeq) that are easily overlooked. Using a central function ensures that subtle and rare problems like this are fixed everywhere once they are discovered.

6 Overview of refinement development

In the crystallographic context structure refinement means optimization of certain target functions by modifying various model parameters. Depending on several factors (e.g. available data, model quality and size) the model can be parameterized in different ways, as grouped or individual atomic parameters. Individual atomic parameters are coordinates, isotropic or anisotropic ADPs (atomic displacement parameters), and occupancy factors. Grouped parameterizations are rigid body, group ADP, TLS, group occupancy, and overall anisotropic scale factor. All of these except group occupancy refinement are currently implemented in `phenix.refine`.

The most recent version of `phenix.refine` allows automatic refinement of any combination of parameters for any part or combination of parts of the model. To the best of our knowledge this is a unique feature among existing crystallographic software.

To give an example, for a molecule with three chains A, B, and C, the command:

```
phenix.refine model.pdb data.mtz \
  strategy=rigid_body+individual_sites+individual_adp+tls \
  sites.rigid_body="chain A" \
  sites.individual="chain B" \
  adp.tls="chain A" \
  adp.tls="chain C"
```

will perform refinement of:

- chain A as a rigid body
- individual isotropic ADPs for the whole molecule
- individual coordinates for chain B
- TLS parameters for chain A and chain C

More information about `phenix.refine` is available at http://phenix-online.org/download/cci_apps/ .

In the following we will highlight a few selected open-source modules supporting `phenix.refine`.

6.1 Rigid body refinement

The core machinery for rigid body refinement is located in `cctbx_sources/mmtbx/mmtbx/refinement/rigid_body.py`. A typical call is:

```
rigid_body_manager = mmtbx.refinement.rigid_body.manager(
  fmodel                  = fmodel,
  selections              = rigid_body_selections,
  refine_r                = True,
  refine_t                = True,
  convergence_test        = True,
  nref_min                = 1000,
  max_iterations          = 25,
  use_only_low_resolution = False,
  high_resolution         = 2.0,
```

```
  low_high_res_limit      = 6.0,
  max_low_high_res_limit  = 8.0,
  bulk_solvent_and_scale  = True,
  bss                     = bulk_solvent_and_scale_parameters,
  euler_angle_convention  = "xyz",
  log                     = log)
rotations    = rigid_body_manager.total_rotation,
translations = rigid_body_manager.total_translation
```

This performs L-BFGS minimization of a crystallographic target w.r.t. `6*len(rigid_body_selections)` parameters. The model (`xray_structure`), crystallographic data (Fobs, ...), and target definition are held by `fmodel`. The selections can cover either the whole molecule or selected parts. Atoms that are not selected are fixed during refinement. Depending on the parameters, it can perform either conventional rigid body refinement in a selected resolution range or use more sophisticated multi-zone protocol where the refinement starts in low resolution zone (defined by `nref_min`) and proceeds with the whole set of reflections. Bulk solvent parameters and scale parameters are updated automatically if the model is shifted more than a certain threshold.

6.2 Grouped isotropic ADP refinement

The code for grouped isotropic ADP refinement resides in `cctbx_sources/mmtbx/mmtbx/refinement/group_b.py`. A typical call is:

```
mmtbx.refinement.group_b.manager(
  fmodel                      = fmodel,
  selections                  = group_adp_selections,
  convergence_test            = True,
  max_number_of_iterations    = 25,
  number_of_macro_cycles      = 3,
  run_finite_differences_test = False,
  log                         = log)
```

This performs refinement of one isotropic ADP per selected group. Non-specific input parameters are similar to those in rigid body refinement module. The refinable parameters are a shift in isotropic ADP for each group, which are added to the original ADPs. The group-specific shifts are applied to both isotropic and anisotropic atoms. For the latter the shift is added to the three diagonal elements of the ADP tensor. In particular this is important for TLS refinement where the atoms in TLS groups are anisotropic and the group ADP refinement is used. ADPs of non-selected atoms are unchanged.

6.3 TLS refinement

This is the most complex code mentioned here and is located in the directories `cctbx_sources/mmtbx/mmtbx/tls` and `cctbx_sources/mmtbx/tls` (Python and C++ code, respectively). The file `cctbx_sources/mmtbx/mmtbx/tls/tools.py` contains the class:

```
class tls_refinement(object):
   def __init__(self, fmodel,
                      model,
                      selections,
                      refine_T,
                      refine_L,
                      refine_S,
                      number_of_macro_cycles,
                      max_number_of_iterations,
                      start_tls_value = None,
                      run_finite_differences_test = False,
                      eps = 1.e-6,
                      out = None,
                      macro_cycle = None):
```

which performs all principal steps in its constructor, including extraction of start TLS parameters from the current ADPs (extracted from `fmodel.xray_structure`) and the current TLS parameters (zero in the first macro-cycle), L-BFGS minimization of a crystallographic target w.r.t. the TLS parameters, split of total ADPs into local and TLS components, and enforcement of positive-definiteness of the final ADP tensors for each individual atom. These operations are exposed as helper functions in `tools.py` which can also be called individually.

6.4 Individual coordinates, ADP and occupancies

The file `cctbx_sources/mmtbx/mmtbx/refinement/minimization.py` is one of the most matrue files in the mmtbx and is the main driver for restrained refinement of individual coordinates, ADPs (isotropic or anisotropic) and occupancies for selected atoms using X-ray and/or neutron data. E.g. to perform coordinate refinement:

```
mmtbx.refinement.minimization.lbfgs(
  restraints_manager       = restraints_manager,
  fmodel                   = fmodel,
  model                    = model,
  refine_xyz               = True,
  lbfgs_termination_params = lbfgs_termination_params,
  wx                       = xray_term_weight,
  wc                       = geometry_term_weight,
  verbose                  = 0)
```

The `model` object contains selection information to determine which atoms are refined and which are fixed.

7 Acknowledgments

We gratefully acknowledge the financial support of NIH/NIGMS under grant number P01GM063210. Our work was supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231.

8 References

Dayhoff, M.O. (1978). Atlas of Protein Sequence and Structure, Vol. 5 suppl. 3, 345-352.
Gotoh, O. (1982). J. Mol. Biol. 162, 705-708.
Grosse-Kunstleve, R.W., Adams, P.D. (2003). Newsletter of the IUCr Commission on Crystallographic Computing, 1, 28-38.
Grosse-Kunstleve, R.W., Afonine, P.V., Adams, P.D. (2004). Newsletter of the IUCr Commission on Crystallographic Computing, 4, 19-36.
Grosse-Kunstleve, R.W., Afonine, P.V., Sauter, N.K., Adams, P.D. (2005). Newsletter of the IUCr Commission on Crystallographic Computing, 5, 69-91.
Henikoff & Henikoff (1992). PNAS 89, 10915-10919
Kabsch, W. (1976). Acta Cryst. A32, 922-923.
Kabsch, W. (1978). Acta Cryst. A34, 827-828.
Kearsley, S.K. (1989). Acta Cryst. A45, 208-210.
Needleman, S. & Wunsch, C. (1970). J. Mol. Biol. 48(3), 443-53.
Smith, T.F. & Waterman M.S. (1981). J. Mol. Biol. 147, 195-197.

An integrated three-dimensional visualization system VESTA using wxWidgets

Koichi Momma[1] and Fujio Izumi[2]
[1] Institute of Mineralogy, Petrology, and Economic Geology, Tohoku University, Aoba, Sendai, Miyagi 980-8578, Japan; [2] Quantum Beam Center, National Institute for Materials Science, 1-1 Namiki, Tsukuba, Ibaraki 305-0044, Japan.
E-mail: monmakou@ganko.tohoku.ac.jp *- WWW:* http://www.geocities.jp/kmo_mma/

1. Introduction

Progress in modern structure-refinement techniques of the maximum-entropy method (MEM) and MEM-based pattern fitting (MPF) [1], has made it easier and more popular to determine three-dimensional (3D) distribution of electron densities from X-ray diffraction data and densities of coherent-scattering lengths (nuclear densities), b_c, from neutron diffraction data. On the other hand, rapid developments of computer hardware and software have accelerated and facilitated electronic-structure calculations affording physical quantities including electron densities, wave functions, and electrostatic potentials.

Such technological advances in recent years bring demands for integrated 3D visualization systems to deal with both structural models and 3D pixel data such as electron and nuclear densities. The crystal structures and spatial distribution of various physical quantities obtained experimentally and by computer simulation should be understood three-dimensionally. Despite the availability of many structure-drawing programs, cross-platform free software capable of visualizing both crystal and electronic structures in three dimensions is very few; if any, they are not very suitable for displaying those of inorganic and metallic compounds.

To improve such a situation, we have recently developed a new integrated system VESTA (**V**isualization for **E**lectronic and **ST**ructural **A**nalysis) for 3D visualization of crystal structures and pixel data on personal computers. This article at first gives a brief overview of VESTA, which is followed by more detailed descriptions of features and algorithm that have been newly implemented in it.

2. Circumstances behind the development of VESTA

VESTA is a successor to two 3D visualization programs, VICS and VEND, in the VENUS (**V**isualization of **E**lectron/**NU**clear densities and **S**tructures) software package [1-4]. VENUS, which was developed by Dilanian, Izumi, and Kawamura with help from Ohki and Momma during 2001–2006, comprises the following five programs:

1. VICS (**VI**sualization of **C**rystal **S**tructures) for displaying and manipulating crystal structures,
2. VEND (**V**isualization of **E**lectron/**N**uclear **D**ensities) for displaying and manipulating 3D pixel data,
3. PRIMA (**PR**actice **I**terative **M**EM **A**nalyses) for MEM analysis from X-ray and neutron diffraction data,
4. ALBA (**A**fter **L**e **B**ail **A**nalysis) for the maximum-entropy Patterson method,
5. Alchemy: a file converter to make it possible to analyze observed structure factors, which result from Rietveld analysis using GSAS and FullProf, by MEM with PRIMA.

VICS and VEND are GLUT- and GLUI-based applications written in the C language with full use of OpenGL technology. They saw the light of day at the end of 2002 and, since then, continued their growth to be used widely in a variety of studies.

However, we never get full satisfaction from their usability and performance for the following four reasons. First, the combined use of them to visualize both crystal and electronic structures *via* text files is rather troublesome; on-the-fly visualization of these two kinds of images is highly desired. Second, their graphical user interface (GUI) is not very user-friendly because they are based on the old-fashioned toolkits, GLUT and GLUI, which have been no longer upgraded. GLUT and GLUI offer only a limited number of widgets. For example, they do not support pull-down menus and tabs for windows, which are, at present, very popular in applications with GUI. One of the most serious troubles in VICS and VEND is that they terminate suddenly without any warning messages whenever the close button at the upper right corner of the title bar in each window/bar has been clicked. This problem results from a bug in the toolkits. Third, VICS and VEND can deal with only a limited number of objects such as atoms, bonds, polyhedra, and polygons on isosurfaces. Fourth, they require large system resources owing to unrefined programming.

To overcome the above faults in VICS and VEND, we at first upgraded VICS to VICS-II employing a modern C++ GUI framework wxWidgets [5,6] to build a new state-of-art GUI and further integrated VICS-II and VEND into the next-generation 3D visualization system VESTA, adding new capabilities.

3. Overview

VESTA is a 3D graphic application written in the C++ language on the basis of OpenGL technology. It runs fast on personal computers equipped with video cards accelerating OpenGL. Windows, Mac OS X, and Linux versions are available. For each platform, both 32- and 64-bit applications will be distributed on the Web.

Thanks to wxWidgets, we can open multiple files using tabbed graphic windows; pull-down menus and tabbed dialog boxes are also supported. Needless to say, the annoying bug related to the close button described above has now been fixed. VESTA allows us to deal with a practically unlimited number of objects as far as memory size goes. It requires much less system resources than VICS and VEND.

VESTA represents crystal structures as ball-and-stick, space-filling, polyhedral, wireframe, stick, and thermal-ellipsoid models. Ball-and-stick, wireframe, and stick models can be overlapped with dotted surfaces corresponding to van der Waals radii. Polyhedra may be made translucent so as to make inside atoms and bonds visible. We can insert a movable lattice plane with variable opacities into a structural model. Drawing boundaries can be defined by ranges along x, y, and z axes as well as lattice planes.

Electron/nuclear densities, wave functions, and electrostatic potentials are visualized as isosurfaces, bird's-eye views, and two-dimensional (2D) maps. VESTA has a feature of surface colorization to show another kind of a physical quantity at each point on isosurfaces. Translucent isosurfaces can be overlapped with a structural model.

VESTA can read in files with 36 kinds of formats such as CIF, ICSD, and PDB and output files with 11 kinds of formats such as CIF and PDB. Users of RIETAN-FP [7,8] must be pleased to learn that standard input files, *.ins, can be both input and output by VESTA. The entire crystal data and graphic settings can be saved in a small text file, *.vesta, without duplicating huge 3D pixel data. File *.vesta with the VESTA format contains relative paths to 3D data files and optionally a crystal-data file that are read in automatically when *.vesta is reopened. VESTA also makes it possible to export graphic files with 14 image formats including 4 vector-graphic ones.

4. Programming models

4.1 GUI parts

wxWidgets [5,6], which was formerly referred to as wxWindows, is one of the best toolkits for cross-platform GUI programming. It provides us with a look-and-feel inherent in each operating system. The

license agreement of wxWidgets, an LGPL-like license with some exceptions allowing binary distribution without source code and copyright, is flexible enough to permit us to develop any types of applications incorporating wxWidgets

During the course of GUI reconstruction, we carefully separated the source code of GUI parts from that of other core parts to make it easier to reuse the latter in combination with other GUI toolkits. The core parts are basically controlled from the GUI ones. However, there are exceptions where functions provided by wxWidgets are called in some core parts. In such a case, the function is wrapped by another function to make the core parts quite independent of GUI toolkits and to show which functions depend on external libraries.

VESTA supports multiple windows, each of which may contain multiple tabs corresponding to files. Various kinds of information is output to the bottom area, where comments can also be input after clicking the comment tab.

Figure 1 shows the main window of VESTA running on Windows XP. A brief hierarchy of widget classes to build it is illustrated in Fig. 2. An OpenGL canvas is placed on a `wxNotebook` widget giving the 'tab' interface. Notebook widgets are usually designed to handle multiple-window components in the same window area. However, we used some tricks to minimize system resources. VESTA was designed in such a way that all the notebook pages have the same kind of a child, *i.e.*, a `wxGLCanvas` widget. Because the notebook widget ensures that no multiple pages are simultaneously displayed, we actually need no multiple OpenGL canvases. Hence, VESTA assigns a single `wxGLCanvas` recursively to every notebook page.

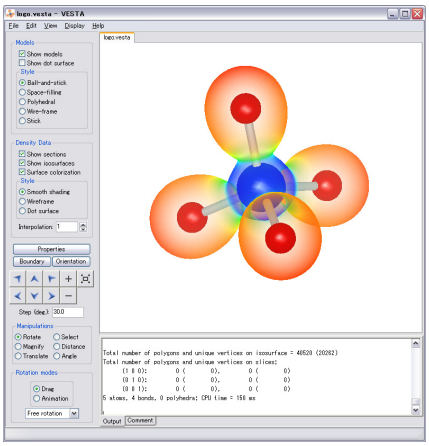


Fig 1: *A screenshot of the VESTA main window.*

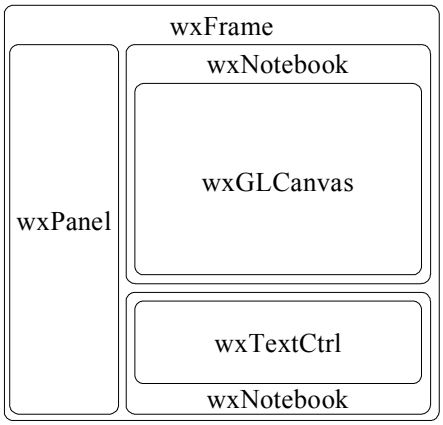


Fig 2: *Schematic image of widgets-class hierarchy on the main window.*

To maximize graphic performance, VESTA uses a cache mechanism of OpenGL display command lists that increase in size with increasing number of drawing objects in the canvas; a command list for only one canvas is always created. If each tab had its own canvas, the total size of memory consumed for the command lists of all the OpenGL canvases would become huge. The layout of widgets in VESTA, therefore, reduces memory usage dramatically. For the Linux and Mac OS X ports, some additional tweaks are required to reuse the same OpenGL canvas in different tabs.

4.2 Core parts

In contrast to the GUI framework, we tried to reuse other parts of the source code as much as possible. All the global variables related to crystal data and graphic settings are capsulated into a `Scene` class to allow object-oriented programming. An instance of the `Scene` class is dynamically generated whenever new data are created or read in from files so that multiple windows and tabs may work simultaneously. The previous programs, VICS and VEND, can deal with only a limited number of objects and consume a large amount of memory even when handling a relatively small number of data because both of them use native ANCI C arrays. To eliminate this limitation with minimum effort and without any appreciable overhead, C arrays were replaced with a small wrapper class of `std::vector`. The main part of the wrapper class is coded as follows:

```
template<typename T> class objVector
{
public:
	~objVector(){
		clear();
	}
	inline T& operator[](size_t i) {
		return (T&)*(T*)v[i];
	}
	inline void add(T* item){
		v.push_back( item );
	}
	void clear(){
		for(size_t i=0; i<v.size(); i++) delete (T*)v[i];
		v.clear();
	}
	void remove(size_t i){
		delete (T*)v[i];
		v.erase(v.begin()+i);
	}
	std::vector<void*> v;
};
```

We also prepared the same kind of a wrapper class, `aryVector`, for pointers to arrays. This class permits type-safe programming without unnesessary code duplication of the `std::vector` template class. Further, it can be accessed by a `[]` operator in exactly the same way as with C arrays on a source-code level despite great differences in real operations on a binary level. We should only note that all the objects must be deletable; that is, they should be generated by a `new` operator. Because they are automatically deleted by a destructor when wasting an array, we need not concern about memory leaks. We have another genuine reason why the `std::vector` class is indirectly used in nearly all cases. Instances of objects can be shared by two or more arrays of the same form. Elements in an array can be quickly sorted only by manipulating pointers without copying a large amount of data for real objects.

5. Dealing with structural models

5.1 Searching for bonds

Bond-search algorithm was much improved in VESTA. The most primitive way to search for bonds is to examine the entire pairs of atoms within drawing boundaries. We adopted a similar approach with some improvements in VICS. Although this algorithm is very simple, the calculation time increases drastically

with increasing number of atoms, N, in the drawing region. The calculation cost is proportional to the square of N, which is expressed as $O(N^2)$ with *big O notation*.[12]

On the other hand, the calculation time is linearly proportional to N, $O(N)$, in VESTA as a result of checking only atom pairs where distances between them are smaller than a certain value. To minimize the number of atoms to be examined, a variant of the cell index method by Quentrec and Brot [9,10] was adopted in VESTA. This approach is widely used in programs for molecular dynamics simulation that needs to deal with a large number of atoms [11]. The basic idea of the cell index method is to divide a large box into subboxes (subcells in the present case) and make up a list of atoms in each subcell. When drawing a crystal structure, the large box corresponds to drawing boundaries, and sizes of a subcell along x, y, and z directions are chosen so as to be slightly larger than the largest bond length. This manner ensures that all the bonds for an atom can be found within the same cell plus neighboring 26 subcells, each of which typically contains only a few atoms.

VESTA can search for bonds in three modes. In the first mode, both *A1* and *A2* atoms bonded to each other are specified. The second mode is used to search for all the atoms located between minimum and maximum distances from an *A1* atom. In the third "Search molecule" mode, a set of atoms within a specified bond length is reiteratively found. Atoms *A1* and *A2* may be specified by entering either symbols for elements or site names.

5.2 Changes in drawing boundaries

Drawing boundaries for structural models can be changed in sophisticated ways similar to convoluting and reiterative-convoluting spheres in ORTEP-III [12]. For example, we usually prefer that coordination polyhedra are not omitted but drawn when their central atoms are situated within the drawing boundaries. For this purpose, all the atoms bonded to these central atoms are added with option "Beyond the boundary" (Fig. 3). To draw Fig. 3, bonds were searched in two different ways. One was to use the "Search A2 bonded to A1" mode with A1 = Al and A2 = O and check "Show polyhedra" and "Search beyond the boundary" options. The minimum and maximum limitations of bond lengths were 0 Å and 2.1 Å, respectively. The other was to use the "Search molecules" mode and apply the "Search beyond the boundary" option to O–H bonds. Bond lengths were set between 0 Å and 1.0 Å, and drawing boundaries between (0, 0, 0) and (1, 1.5, 1).

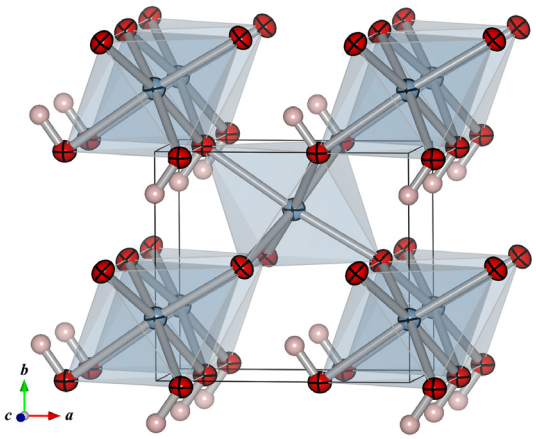


Fig 3: *Crystal structure of δ–AlOOH, a high pressure modification of aluminum oxide hydroxide, with thermal-displacement ellipsoids at a 99 % probability level [13]. H atoms are represented by spheres of an arbitrary radius.*

[12] The big *O* notation is a mathematical one to describe the behavior of a function for a very large (or a very small) number of input data, in this case, N atoms. More precisely, "$f(n)$ is $O(g(n))$" means that there exists a certain constant m and a positive constant c such that $f(n) \leq c*g(n)$ for $n > m$.

In organic compounds including metal complexes, the combination of the "Search molecule" mode and option "Beyond the boundary" is used to search for the entire atoms of molecules with at lease one atom lying within drawing boundaries.

5.3 Information about objects

Selection of objects (atoms, bonds, and coordination polyhedra) by clicking with a mouse provides us with a variety of crystallographic information in the output window at the bottom:

1) fractional coordinates,
2) symmetry operations and translation vectors,
3) site multiplicities plus Wyckoff letters derived by STRUCTURE TIDY [14] embedded in RIETAN-FP (for files of some formats)
4) interatomic distances, bond angles, and torsion angles,
5) information about coordination polyhedra including volumes, Baur's distortion indices [15], quadratic elongations [16], bond angle variances [16], bond valence sums [17] of central metals, and bond lengths expected from bond valence parameters [17].

5.4 Lattice transformation

VESTA has a feature to convert general equivalent positions in a conventional setting into those in a non-conventional one with a transformation matrix, which is also used for (primitive lattice)–(complex lattice) conversions and for creating superstructures.

The (4×4) transformation matrix $\mathbb{P}=\begin{pmatrix}\boldsymbol{P} & \boldsymbol{p}\\ 0 & 1\end{pmatrix}$ consists of the (3×3) rotation matrix $\boldsymbol{P}$ and the (3×1) translation vector $\boldsymbol{p}$. Primitive translation vectors $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ are transformed by $\boldsymbol{P}$ as

$$\begin{aligned}(\boldsymbol{a}',\boldsymbol{b}',\boldsymbol{c}') &= (\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\boldsymbol{P}\\ &= (\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\begin{pmatrix}P_{11} & P_{12} & P_{13}\\ P_{21} & P_{22} & P_{23}\\ P_{31} & P_{32} & P_{33}\end{pmatrix}\\ &= (P_{11}\boldsymbol{a}+P_{21}\boldsymbol{b}+P_{31}\boldsymbol{c},\ P_{12}\boldsymbol{a}+P_{22}\boldsymbol{b}+P_{32}\boldsymbol{c},\ P_{13}\boldsymbol{a}+P_{23}\boldsymbol{b}+P_{33}\boldsymbol{c}).\end{aligned}$$

The shift of the origin is defined with the shift vector

$$\mathbf{p}=(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\boldsymbol{p}=(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\begin{pmatrix}p_1\\ p_2\\ p_3\end{pmatrix}=p_1\boldsymbol{a}+p_2\boldsymbol{b}+p_3\boldsymbol{c}\,.$$

The symmetry operation $\mathbb{W}$ is transformed by

$$\mathbb{W}' = \mathbb{Q}\mathbb{W}\mathbb{P},$$

where $\mathbb{Q}=\begin{pmatrix}\boldsymbol{Q} & \boldsymbol{q}\\ 0 & 1\end{pmatrix}=\begin{pmatrix}\boldsymbol{P}^{-1} & -\boldsymbol{P}^{-1}\boldsymbol{p}\\ 0 & 1\end{pmatrix}=\mathbb{P}^{-1}.$

The lattice vector $\mathbb{x}=\begin{pmatrix}x\\ y\\ z\\ 1\end{pmatrix}$ is transformed by

$$\mathbb{x}' = \mathbb{Q}\mathbb{x}.$$

If the determinant of ***P***, det(***P***), is negative, the crystal-coordinate system is transformed from right-handed to left-handed, and *vice versa*. If det(***P***) is not 1, the unit-cell volume, *V*, changes on the lattice transformation. When det(***P***) is larger than 1, VESTA allows us to create a superstructure by examining the following equation to find additional sites lying in between (0,0,0) and (1,1,1).

$$\begin{pmatrix} x' \\ y' \\ x' \\ 1 \end{pmatrix} = \mathbb{Q} \begin{pmatrix} x+n_x \\ y+n_y \\ z+n_z \\ 1 \end{pmatrix} \qquad (n_x, n_y, n_z = 0, \pm 1, \pm 2, \ldots).$$

Note that VESTA adds no new equivalent positions but new sites when creating superstructures. That is, the space group of the new structure should be a subgroup of the original space group. When det(***P***) is smaller than 1, the same position may result from two or more sites. In such a case, the redundant sites can be removed by clicking a "Remove duplicate atoms" button.

6. Dealing with 3D pixel data

6.1 Three kinds of representation

Isosurfaces are represented as smooth-shaded polygons, wireframes, and dot surfaces. Physical quantities, *e.g.*, wave functions and nuclear densities, having both positive and negative values are expressed by isosurfaces with two different colors.

6.2 Surface colorization

Another attractive feature is surface colorization, whose typical application is to colorize isosurfaces of electron densities according to electrostatic potentials. Figure 4 exemplifies an application of this feature to results of an electronic-state calculation for the $[Cd\{S_4Mo_3(Hnta)_3\}_2]^{4-}$ ion (H_3nta: nitrilotriacetic acid) [18] by a discrete variational Xα method with DVSCAT [19]. The isosurface level was set at $0.03a_0^{-3}$, and electrostatic potentials on the isosurface range from –0.814 Ry (blue) to 0.174 Ry (red). Opacity parameters, C_1 and C_2, for the isosurface (see 6.4) were 80 % and 100 %, respectively.

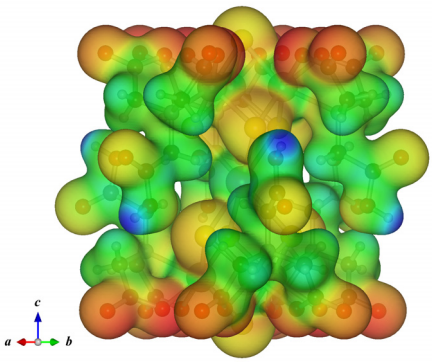


Fig 4: *A composite image of an electron-density isosurface colorized on the basis of electrostatic potentials and a ball-and-stick model for the* $[Cd\{S_4Mo_3(Hnta)_3\}_2]^{4-}$ *ion [18].*

6.3 Two-dimensional distribution on lattice planes and boundary sections

Lattice planes (slices) can be inserted to display 2D distribution of a physical quantity in addition to its isosurfaces. Both boundary sections and lattice planes are colorized on the basis of numerical values on them. The saturation level of color is specified as (a) a value normalized between 0 and 100 corresponding minimum and maximum values, respectively, and (b) a value corresponding to raw data. The color index, T, for a data point with a value of d is calculated from two saturation levels, S_{min} and S_{max}, in the unit of the raw data:

$$T = \frac{d - S_{\mathrm{min}}}{S_{\mathrm{max}} - S_{\mathrm{min}}}.$$

Data points with values larger than S_{max} and smaller than S_{min} are given the same colors as those assigned to S_{max} and S_{min}, respectively. The color of each point is determined from T, depending on color modes: B-G-R, R-G-B, C-M-Y, Y-M-C, gray scale (from black to white), and inversed gray scale. For example, blue is assigned to 0, green to 0.5, and red to 1 in the B-G-R mode.

6.4 How to calculate opacities of polygons

With VESTA, the visibility of both outlines of isosurfaces and an internal structural model was surprisingly improved in comparison with that with VEND. The opacity, $C(p)$, for polygon p with a normal vector of (x, y, z) is calculated with a linear combination of two opacity parameters, C_1 and C_2, for polygons parallel and perpendicular to the screen, respectively:

$$C(p) = C_1 z + C_2(1 - z).$$

6.5 Order of drawing polygons

When drawing translucent objects with the OpenGL API, polygons should be usually drawn from behind. VESTA sorts a set of polygons before drawing them, making it possible to render images properly regardless of orientation of the objects. The sequence of drawing polygons can optionally be reversed. The use of this option may sometimes improve the visibility of complex isosurfaces because neither back surfaces nor internal ones are drawn; otherwise, they heavily overlap each other to give an improper image.

6.6 Pixel operations

VESTA supports pixel operations between more than two 3D data sets. For example, this feature enables us (a) to subtract calculated electron densities from observed ones obtained by MEM analysis to detect atoms missing in a structural model and (b) to subtract down-spin electron densities from up-spin ones to visualize effective spin densities (Fig. 5).

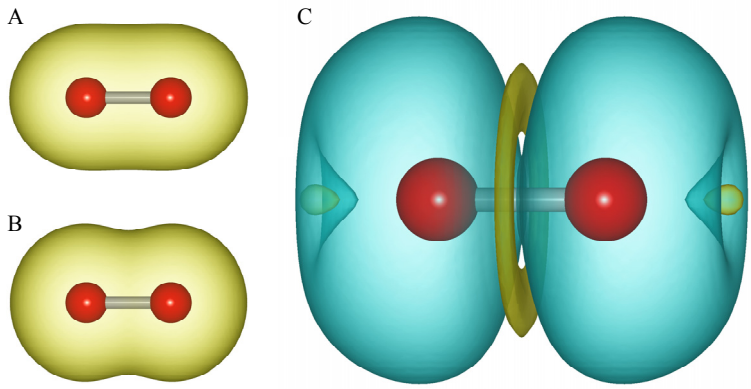


Fig 5: *Distributions of electron densities and effective spin densities calculated with DVSCAT for the O_2 molecule. A: up-spin electron density; B: down-spin electron density; C: effective spin density (A – B) calculated with VESTA. Isosurface levels were set at $0.01a_0^{-3}$ (A and B) and $0.001a_0^{-3}$ (C), respectively.*

6.7 Improvements in rendering quality

The quality of rendering objects such as isosurfaces, boundary sections, and lattice planes in VESTA is much higher than that in VEND, even when dealing with a relatively small number of pixels.

6.7.1 Smooth-shading model

A flat-shading model to draw isosurfaces was used in VEND. It assigns a normal vector to each polygon, which causes the discontinuity of reflection colors near vertices and edges shared with neighboring polygons. To settle this problem, we adopted a smooth-shading model, which assigns a normal vector to each vertex of a polygon, in VESTA. The normal vector at each point on the polygon is calculated with a shading model, 'GL_SMOOTH', provided by the OpenGL API using a linear combination of the normal vectors at the vertices.

6.7.2 Colors for boundary sections and lattice planes

VEND assigns RGB values to vertices of polygons on boundary sections and lattice planes. However, these 'colored polygons' are problematic when data values mainly change in a narrow region. Suppose a situation where blue is assigned to 0, green to 0.5, and red to 1. If the three vertices of a triangular polygon have values of 0, 0.5, and 1, the color for the center of the polygon becomes a mixture of the three colors:

$$R = (0 + 0 + 1)/3 = 1/3$$
$$G = (0 + 1 + 0)/3 = 1/3$$
$$B = (1 + 0 + 0)/3 = 1/3.$$

The resultant color is gray that does not represent any physical quantities. To avoid the appearance of such a meaningless color, VESTA prepares a rainbow-colored one-dimensional texture, glTexImage1D, and assigns coordinates in the texture to vertices. For RGB texture, 0 is assigned to blue texture positions, 0.5 to green ones, and 1 to red ones. Then, for any sets of three physical quantities at vertices and for any positions of polygons, texture coordinates represent values at those positions. In the above case, the texture coordinate (= color index), T, at the center of the polygon is:

$$T = (0 + 0.5 + 1)/3 = 0.5.$$

The internal unit of saturation levels for colors is changed from a normalized one to a raw-data one so that saturation levels close to 0 or 1 (in a normalized unit) can work correctly. Assume that we specify saturation levels normalized between 0 and 1 corresponding to minimum and maximum values in a data set, respectively. Then, a problem arises from floating-point arithmetic errors if 3D data have extremely large (or small negative) values as in the case of electrostatic potentials. For instance, suppose a case where minimum and maximum data values are respectively -10^{20} and 0 with saturation levels from –10 to 0 in the raw-data unit. The normalized value for –10 is $(-10 + 10^{20})/(0 + 10^{20}) \approx 1$ while that for 0 is $(0 + 10^{20})/(0 + 10^{20}) = 1$. Thus, with the normalized representation, values between –10 and 0 cannot be distinguished within the precision of floating-point arithmetic. This problem does not occur in VESTA because the internal representation of saturation levels, –10 and 0 in the above case, has the same precision as with raw data. Thus, VESTA generates no factitious lines on colored isosurfaces.

6.7.3 Surface colorization

For colorization of isosurfaces, unrefined code was written in VEND. That is, data values at vertices on the isosurfaces are approximated at those of the nearest pixels while different values are assigned to those vertices of neighboring polygons which have just the same coordinates. We thoroughly redesigned the surface-colorization feature in VESTA. With VESTA, the data value, $U(v)$, for vertex v between two pixels, p_1 and p_2, is evaluated using a linear combination of $U(p_1)$ and $U(p_2)$:

$$U(v) = \frac{U(p_1)l_2 + U(p_2)l_1}{l_1 + l_2}$$

where l_1 and l_2 are the distance between v and p_1, and that between v and p_2, respectively; $l_1 + l_2$ is equal to the distance between p_1 and p_2. Thus, VESTA shows much smoother isosurfaces without interpolation of pixel data than VEND.

6.8 Algorithm for geometry calculations

The calculation of isosurface geometry has been appreciably accelerated in VESTA compared with that in VEND.

Figure 6 shows a 2D projection of 3D pixel data and 'grids' defined by pixel rows along *x*, *y*, and *z* axes. Vertices and polygons on isosurfaces are calculated for all the subgrids with eight pixels per grid. For each subgrid, its twelve edges are checked whether they intersect the isosurfaces or not. For example, if a pixel data, $p(i,j,k)$, is larger than the isosurface level whereas $p(i\text{-}1,j,k)$ is smaller than it, a vertex between $p(i,j,k)$ and $p(i\text{-}1,j,k)$ for a polygon is generated. Every polygon on the isosurface generated in this way has its own three vertices. However, all the pixels except those on drawing boundaries are shared with neighboring eight subgrids, which means that values at most pixels are evaluated as many as eight times. Furthermore, this method generates many duplicate vertices with the same coordinates because most vertices on the isosurface are common among neighboring six polygons. The sluggish speed of calculating isosurface geometry with VEND is explained in terms of these facts. The simple but inefficient approach to calculate isosurface geometry is given below:

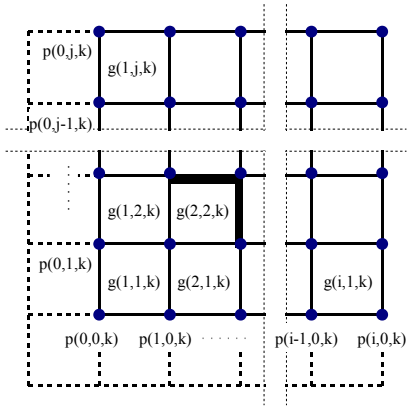


Fig 6: *The relation between 3D pixel data* ***p*** *and grid* ***g*** *defined by nearest eight pixels. Polygons for the isosurface are calculated for each grid.*

```
calculate_isosurface_geometry_old(
        float *rho,    // 3D pixel data
        float isolevel  // isosurface level
        int   bbox[6], // Drawing boundaries
        int   NPIX[3], // Number of pixels along x, y, z in unit cell
        ){
        int   i, j, k, m;
        int   r[8][3];
        float grid[8][4];
        float *vertlist[12];
        objVector<Polygon> poly; // list of polygons on isosurface
        unsigned char cubeindex; // bit flags for a grid recording which pixels are outside/inside of
                                        // isosurface
        ...

        r[0][0] = 0; r[0][1] = 0; r[0][2] = 0;
        r[1][0] = 1; r[1][1] = 0; r[1][2] = 0;
        r[2][0] = 1; r[2][1] = 1; r[2][2] = 0;
        r[3][0] = 0; r[3][1] = 1; r[3][2] = 0;
        r[4][0] = 0; r[4][1] = 0; r[4][2] = 1;
        r[5][0] = 1; r[5][1] = 0; r[5][2] = 1;
        r[6][0] = 1; r[6][1] = 1; r[6][2] = 1;
        r[7][0] = 0; r[7][1] = 1; r[7][2] = 1;

        for (k=bbox[4]; k<bbox[5]; k++){
                for (j=bbox[2]; j<bbox[3]; j++){
                        for (i=bbox[0]; i<bbox[1]; i++){
```

```
                        for (m=0; m<8; m++){
                                int id = pos(i, j, k); // pos(i, j, k) reterns integer index of pixel(i, j, k)
                                grid[m][0] = i + r[m][0];
                                grid[m][1] = j + r[m][1];
                                grid[m][2] = k + r[m][2];
                                grid[m][3] = rho[id];
                                if (rho[id] < isolevel) cubeindex |= 1 << m;
                        }
                        // All the eight pixels are evaluated. Now calculate vertices for tweleve edges.
                        // 'edgeTable' is a list of IDs used to judge wether the edge intersects the
isosurface or not.
                        // calc_vtx() returns coordinates of a vertex.
                        if(edgeTable[cubeindex] & 1)    vertlist[0]  = calc_vtx(isolevel, grid[0], grid[1]);
                        ...
                        if(edgeTable[cubeindex] & 1024) vertlist[10] = calc_vtx(isolevel, grid[2], grid[6]);
                        if(edgeTable[cubeindex] & 2048) vertlist[11] = calc_vtx(isolevel, grid[3], grid[7]);

                        // Build polygons using vertices stored in vertlist.
                        ...
                } // end loop(i)
        } // end loop(j)
    } // end loop(k)
}
```

What is worse, if we try to render an isosurface with the smooth-shading model, redundant vertices of the isosurface should be reduced to a single vertex to calculate a normal vector. Such a situation should be avoided because of the following three reasons: pixel values are evaluated as many as eight times per pixel; vertices on the isosurface are generated six times greater than needed; a large number of vertices generated in this way are compared with each other to check duplication.

Our new algorithm introduced into VESTA eliminates all of these redundant calculations by a refinement of the order of loops, as can be appreciated from Fig. 6.

```
calculate_isosurface_geometry_new(
        float *rho,       // 3D pixel data
        float  isolevel   // isosurface level
        int  bbox[6],    // Drawing boundaries
        int  NPIX[3],    // Number of pixels along x, y, z in unit cell
        ){
        int i, j, k;
        aryVector<float*> vtx;     // list of vertices on isosurface
        unsigned char *cubeindex; // bit flags for each grid recording which pixels are outside/inside of
                                        // isosurface
        long *edgeList;          // edge tables recording integer identifier of vertices on the edges
        ...
        cubeindex = new unsigned char[2*(bbox[1]-bbox[0] +1)* (bbox[3]-bbox[2]+1)];
        edgeList = new long[6*(bbox[1]-bbox[0] )* (bbox[3]-bbox[2])]

        for (k=bbox[4]; k<=bbox[5]; k++){
                ...     // copy cubeindex(i,j,k) to cubeindex(i,j,k-1), reset cubeindex(i,j,k).
                        // copy edgeList(i,j,k) to edgeList(i,j,k-1).
                for (j=bbox[2]; j<=bbox[3]; j++){
                        for (i=bbox[0]; i<=bbox[1]; i++){
                                int M = pos(i, j, k); // pos(i, j, k) reterns integer index of pixel(i, j, k)
                                if (rho[M] > isolevel){
                                        ...
                                        // register bit flags to 8 grids around this pixel that this pixel is inside
                                        // of isosurface
                                }
                                if(i > bbox[0]){
                                        ...
                                        // Calculate vertices between p(i-1, j, k) and p(i, j, k),
                                        // add to vtx and register it to edgeList.
```

```
                    }
                    if(j > bbox[2]){
                        ...
                        // Calculate vertices between p(i, j-1, k) and p(i, j, k),
                        // add to vtx and register it to edgeList.
                    }
                    if(k > bbox[4]){
                        ...
                        // Calculate vertices between p(i, j, k-1) and p(i, j, k),
                        // add to vtx and register it to edgeList.
                    }

                    if(i > bbox[0] && j > bbox[2] && k > bbox[4]){
                        ...
                        // Generate polygons for grid(i, j, k) using the bit flag cubeindex(i, j, k)
                        // and edgeList.
                    }
                } // end loop(i)
            } // end loop(j)
        } // end loop(k)
}
```

Note that `cubeindex` is now changed to an array of indices for two sheets with k=k-1 and k=k in grids arranged three-dimensionally. Instead of carrying out a loop for grids to check pixel values, our new algorithm directly carries out loops for pixels and updates the indices of eight grids around the pixel in every cycle. In the same cycle, the coordinates of vertices on the three edges of grid $g(i,j,k)$ are calculated and stored in an array `vtx`. For example, in cycle (i=2, j=2, k=k), the coordinates of vertices between $p(1,2,k)$ and $p(2,2,k)$, $p(2,1,k)$ and $p(2,2,k)$ (bold lines in Fig. 6), and $p(2,2,k$-1) and $p(2,2,k)$ are calculated. Other vertices of grid $g(2,2,k)$ have already been calculated in other cycles, (i=1,j=2,k=k), (i=2,j=1,k=k), *etc*. Then, in the same cycle, polygons for grid $g(2,2,k)$ are generated using a list of vertices stored in `vtx`.

This algorithm has another advantage that the memory size for data at vertices is significantly reduced owing to the reuse of data at vertices among neighboring polygons.

7. Other new features

7.1 Cutoff planes

Drawing boundaries are fundamentally specified by inputting ranges along x, y, and z axes in the "Boundary" dialog box, where we can further specify "Cutoff planes". Each cutoff plane is specified with Miller indices h, k, and l, and a central distance from the origin (0,0,0). After atoms, bonds, and isosurfaces within the x, y, and z ranges have been generated, those lying outside of the cutoff planes are excluded. Therefore, this feature is particularly useful to visualize distribution of 3D pixel data on lattice planes in addition to isosurfaces. The central distance of a cutoff plane may be specified in the unit of either its lattice-plane spacing, d, or Å.

7.2 Hiding objects efficiently

Two or more atoms, bonds, and polyhedra can now be selected with mouse dragging and hidden by pressing the Delete key. Pressing the Esc key resumes the hidden objects.

7.3 Improvement in depth-cueing

Depth-cueing was revised to allow more flexible settings. An OpenGL API, `glFog`, with a 'GL_LINEAR' mode is used to enable depth-cueing. It blends a 'fog' color with the original color of each object using the blending factor f. Both of the starting depth, *start*, and the ending depth, *end*, are defined by users. The factor f at depth z is computed by

$$f = \frac{end - z}{end - start}.$$

VESTA automatically assigns the background of the canvas to the fog color *Cf*. Then, the color, *Cr*, of a rendering object is replaced by

$$Cr' = f * Cr + (1 - f) * Cf \ .$$

When VESTA displays objects on the OpenGL canvas, the depth of each object, *z*, is normalized in such a way that the radius of a bounding sphere for the scene becomes 0.9. Then, *start* = 0.0 and *end* = 1.0, for example, clearly render the half of the scene closer than *z* = 0.0 without any depth-cueing whereas the farthest objects (*z* = 0.9) are completely invisible.

8. Collaboration with external programs

8.1 ORFFE

ORFFE [20] calculates geometrical parameters from data in file *.xyz created by RIETAN-FP, outputting them in file *.ffe. When reading in *.lst and/or *.ins, VESTA also inputs *.ffe automatically provided that *.ffe shares the same folder with *.lst and/or *.ins. Otherwise, *.ffe can be input by pressing the "Read *.ffe" button in the "Geometrical Parameters" dialog box. This dialog box lists interatomic distances and bond angles recorded in *.ffe.

VESTA allows us to locate the bonds and bond angles displayed in the "Geometrical Parameters" dialog box on a graphic window. On selection of a bond (2 atoms) or a bond angle (3 atoms) in this dialog box, the corresponding object in a ball-and-stick model is highlighted, and *vice versa* (Fig. 7). Thus, atom pairs and triplets associated with geometrical parameters on which restraints are imposed in Rietveld analysis with RIETAN-FP are easily recognized in the ball and stick model.

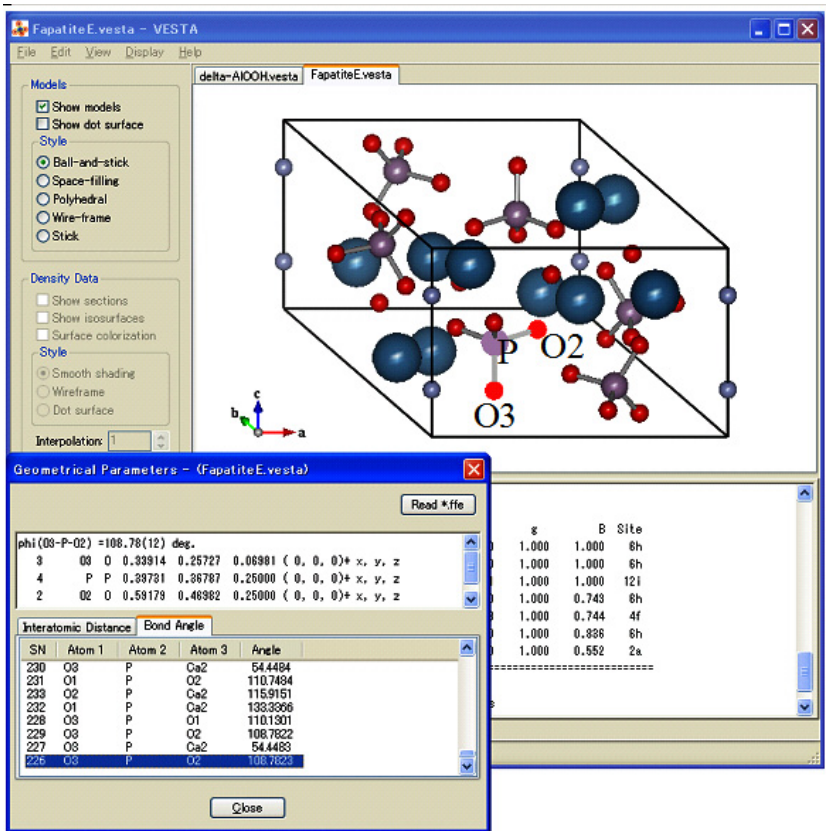


Fig 7: *"Geometrical Parameters" dialog box showing a list of bond angles recorded for fluorapatite in *.ffe. An O3–P–O2 angle is highlighted in the graphic window.*

Because ORFFE calculates estimated standard deviations (e.s.d.'s) of geometrical parameters with both diagonal and off-diagonal terms in the variance-covariance matrix, the resulting values of e.s.d.'s are improved compared with those given by VESTA using only diagonal terms.

8.2 RIETAN-FP

wxWidgets provides us with function `wxExecute` and class `wxProcess` to call external programs easily from existing processes. Therefore, we can virtually extend VESTA by running other applications as a child process, yet keeping its core code simple and small. VESTA utilizes this framework to simulate powder diffraction patterns with RIETAN-FP. On selection of the "Powder Diffraction Pattern…" item under the "Utilities" menu, a series of procedures, *i.e.*, generation of an input file, *.ins, for RIETAN-FP, execution of RIETAN-FP, and graphic representation of the resulting data in file, *.itx, with a graphing program such as Igor Pro and gnuplot, is executed by VESTA as if they were implemented in VESTA.

9. Summary

With all the advanced features and high performance described above, VESTA is expected to contribute many investigations of crystal and electronic structures. It will act as a 'mediator' between structure analyses and electronic-structure calculations. We will polish it up further to make it more useful and more powerful.

References

[1] F. Izumi and R. A. Dilanian, in: Recent Research Developments in Physics, vol. 3, part II, Transworld Research Network, Trivandrum (2002) pp. 699–726,.
[2] F. Izumi: *Rigaku J.*, 36, No. 1 (2005) 18.
[3] F. Izumi and R. A. Dilanian: *Commission on Powder Diffr., IUCr Newslett.,* No. 32 (2005) 59.
[4] F. Izumi and Y. Kawamura: *Bunseki Kagaku*, 55 (2006) 391.
[5] Information obtainable from http://www.wxwidgets.org/
[6] J. Smart, K. Hock, and S. Csomor: Cross-Platform GUI Programming with wxWidgets, Prentice Hall (2005).
[7] F. Izumi and T. Ikeda: *Mater. Sci. Forum*, 321–324 (2000) 198.
[8] F. Izumi and K. Momma: *Proc. XX Conf. Appl. Crystallogr.*, *Solid State Phenom.*, in press.
[9] B. Quentrec and C. Brot: *J. Comp. Phys.*, 13 (1975) 430.
[10] M. P. Allen and D. J. Tildesley: Computer simulation of liquids, Clarendon Press, Oxford (1987) p. 149.
[11] K. Refson: *Computer Phys. Commun*., 126 (2000) 310.
[12] M. N. Burnett and C. K. Johnson: Report ORNL-6895, Oak Ridge National Laboratory (1996).
[13] K. Komatsu, T. Kuribayashi, A. Sano, E. Ohtani, and Y. Kudoh: *Acta Crystallogr., Sect. E*, 62 (2006) 216
[14] L. M. Gelato and E. Parthé: *J. Appl. Crystallogr*., 20 (1987) 139.
[15] W. H. Baur: *Acta Crystallogr.*, *Sect. B*, 30 (1974) 1195.
[16] K. Robinson, G. V. Gibbs, and P. H. Ribbe: *Science*,172 (1971) 567.
[17] I. D. Brown and D. Altermatt: *Acta Crystallogr.*, *Sect. B*, 41 (1985) 244.
[18] G. Sakane, H. Kawasaki, T. Oomori, M. Yamasaki, H. Adachi, and T. Shibahara: *J. Cluster Sci.,* 13 (2002) 75.
[19] H. Adachi, M. Tsukada, and C. Satoko: *J. Phys. Soc. Jpn.*, 45 (1978) 875.
[20] W. R. Busing, K. O. Martin, and H. A. Levy: Report ORNL-TM-306, Oak Ridge National Laboratory (1964).

Visual Graphic Library VGLIB5 for Crystallographic Programs on Windows PCs

Kenji Okada*[a], Ploenpit Boochatum[a], Keiichi Noguchi[b] and Kenji Okuyama[c]
*[a]Faculty of Science, King Mongkut's University of Technology Thonburi, Bangmod, Tungkru, Bangkok 10140, Thailand, [b]Instrumentation Analysis Center, Tokyo University of Agriculture and Technology, Koganei, Tokyo 184-8588, Japan, and [c]Department of Macromolecular Science, Graduate School of Science, Osaka University, 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan; Email: *kokada@kmutt.ac.th, ploenpit.boo@kmutt.ac.th, knoguchi@cc.tuat.ac.jp, okuyamak@chem.sci.osaka-u.ac.jp ; WWW: http://www.ccp14.ac.uk/ccp/web-mirrors/okada/, http://www.chem.sci.osaka-u.ac.jp/lab/tashiro/index.html*

Abstract

VGLIB5 (Visual Graphic Library version 5) is a graphic library for scientific simulation program. The capabilities of the VGLIB5 are: 1) Fortran callable routines (CALCOMP and/or Basic like), 2) Display on MS-Windows (PCs) or X-Windows (HP/IBM/SGI/SUN), 3) Prepare HP-GL 758X plotter files, 4) Prepare Postscript files, and 5) Prepare a VTR tape using IMAGE memory. And the basic concept is a very simple. This library 1) does not depend on hardware (from super-computer to PCs) and OS, 2) is callable from Fortran and/or C languages, 3) can accept the input controls from keyboard and mouse, and output the results to color display, VTR, HP-GL plotter and Postscript printers, and 4) has adopted the layered structure for easy maintenance.

1 Introduction

Several crystallographic subroutine libraries [1] and several graphics user interfaces (GUI) [2] for cross-platform are also reported on IUCr Computing Commission newsletters in detail. Applications of computer graphics are allpied covering polysemy such as Film, Game, Animation, Scientific/ Visualization simulation, Web graphs, Artitecture/Product design and Print/Publishing. These applications are developed and distributed as a software package or a handmade software. To develope a new graphical software in scientific simulation, firstly we must examine the target platforms (mainframes, Workstations, Personal Computers, and graphic terminals) with operating system. Secondary the new software adapts an exsisting and widely used graphic library that has a graphic application programming interface (API). Most graphic libraries are dependent on the platform, and like a graphics processing unit, do not exhibit the interface specification, and have not become open source. Thirdly this software should be implemented widely using programming languages such as Fortran, C/C++ or Visual Basic.

Adopting the graphic library having interface to Fortran and/or C/C++ languages, we once more examine the input/output functions. In scientific simulation, input devices mostly used are digital camera, scanner, OCR (Image/Text), keyboard and mouse, and output devices are display, printer/plotter, video tape recorder (VTR) and/or files. After ACM-SIGGRAPH [3] was established in 1969, there are many graphic software libraries as standards that have 'de-facto' standards for the portability of graphic programs such as Graphical Kernel System (GKS, GKS-3D) [4], Programmers Hierarchical Interactive Graphics System (PHIGS, PEX) [5], PostScript [6], or X Window System (X) [7]. Recently windows based graphic libraries such as OpenGL [8], PGPLOT [9], Gnuplot [10], Alias [11] and AVS [12] are used in MS-Windows and X-Windows of Unix/Linux computers. Several libraries are free of charge.

If the new planned program in scientific simulation is very simple as like as crystallographic computation and uses limited input/output devices such as no input and three outputs (display, plotter/postscript file and VTR), these graphic libraries appeared on the market in the world are very heavy to execute our new program. The essential functions of the graphics library in these simulations are draw/display colored

points, lines, planes, fill area, texts, view-port, and etc. If these functions are callable from Fortran and/or C/C++ languages, it is the very graphic library that we desire.

In 1991, the first version of a graphic library GRLIB (Graphic Library) was proposed for the scientific simulation program using Fortran language [13]. This GRLIB did not depend on any platform such as super-computer (Cray/COS), mainframe (CDC6600/SCOPE) and 32 bits PCs (MS-DOS), and was able to treat most popular plotter devices that time having Calcomp/PLOT10 protocols. The GRLIB2 (version 2) was developed in 1996 [14] adding VTR features using image memory for IBM 5080 display of mainframe (IBM3090/ MVS), workstation (HP900/HP-UX) and PCs (MS-DOS). The VGLIB3 (Visual Graphic Library, version 3) as an implemented version of GRLIB2 was developed in 1997 [15] adding the features of CRT display and HP-GL protocol under MS-Windows PCs (NT 3.x/95). The VGLIB4 (version 4) adopted layer approach and added the Postscript protocol on MS-Windows PCs (NT3.x/4.0/95/98) [16]. In 2005, the VGLIB5 (version 5) was implemented as a unified graphic library for scientific simulation from the VGLIB4. This VGLIB5 written in Fortran90/95 is used in DS5 (DIRECT-SEARCHER Automatic System version 5) for structure analysis of small molecules running on PCs [17], and works on X-Windows (Unix/ Linux). In this paper we focus on the layered structural design and features of DSLIB5.

2 Description

2.1 Basic Concept

The basic concept in the VGLIB5 for scientific simulation is very simple. This visual graphic library 1) does not depend on hardware (from super-computer to PCs) and operating system (OS), 2) is callable from Fortran and/or C languages, 3) can accept the input controls from keyboard and mouse, and output the results to color display, VTR, HP-GL plotter and Postscript printers, and 4) has the layered structure for easy maintenance.

2.2 Capabilities of VGLIB5 (MS-Windows and Unix/Linux):

There are five capabilities: 1) Fortran callable routines (CALCOMP and/or Basic like), 2) Display on MS-Windows (PCs) or X-Windows (HP/IBM/SGI/SUN), 3) Prepare HP-GL 758X plotter file, 4) Prepare Postscript file, and 5) Prepare VTR tape using IMAGE memory.

2.3 Layered Structure

The VGLIB5 has the layered approach (Fig. 1). The bottom layer is the driver groups depending on each device (8 drivers), the second one is common routines of basic primitives and functions, and the third one is application routines according to each purpose. Each routine of three layers can be called directly from the user program.

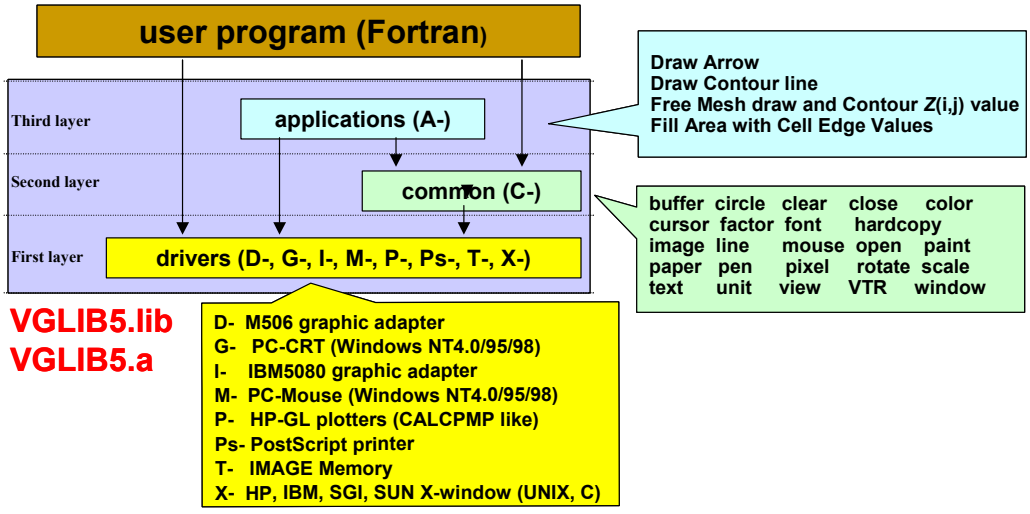


Fig 1: *Layered structure of VGLIB5*

The driver layer consists of eight series as follows. The routines in X-series equal ones in G-series. (just differ the platform)

Series	File	Comment	No. of routines
D-	VGLIBD.for	for M506 graphic adapter	M506, 9
G-	VGLIBG.for	for PC-CRT (Windows)	PCs, 36
I-	VGLIBI.for	for IBM5080 graphic adapter	IBM5080, 2
M-	VGLIBM.for	for PC-Mouse (Windows)	Mouse, 4
P-	VGLIBP.for	for CALCPMP Plotter	Plotter, 13
PS-	VGLIBPS.for	for PostScript printer	Postscript, 10
T-	VGLIBT.for	for IMAGE Memory	IMAGE memory, 10
X-	VGLIBXI.c	for HP, IBM X-window of UNIX Fortran	X-window, ANSI C, 36
	VGLIBXS.c	for SGI, SUN X-window of UNIX Fortran	X-window, ANSI C, 36

The common layer has three series used in commonly such as text, for the display and paper handling.

Series	File	Comment	No. of routines
C-	VGLIBC1.for	Text data	alphabetic, numeric, symbols, 1
	VGLIBC2.for	VGxxx routines	display control, 20
	VGLIBC3.for	set CRT/Paper	view port, paper size, etc., 4

The application layer has the collection of the routines with a certain purpose.

Series	File	Comment	No. of routines
A-	VGLIBA.for	application	arrow, contour, grading, etc., 4

2.4 Colors and Fonts

Actual colors and color codes of MS- and X-Windows are quite different. The VGLIB5 unified these color codes from 0 to 16 as follows in MS-Windows: 0(black), 1(light blue), 2(light red), 3(light magenta), 4(light green), 5(light cyan), 6(yellow), 7(bright white), 8(gray), 9(blue), 10(red), 11(magenta), 12(green), 13(cyan), 14(brown), 15(white), and 16(gray). When user want use more than sixteen colors, user must select one within these 16 colors. Fonts used in MS-Windows are restricted to eight kinds: 1(Curie), 2(Helv.), 3(tms rms), 4(modern), 5(script), 6(roman), 7(system), and 8(preview). In X-windows, color codes are: 0(black), 1(blue), 2(red), 3(magenta), 4(green), 5(cyan), 6(yellow), 7(white), 8(light gray), 9(coral), 10(slate blue), 11(orange), 12(spring green), 13(blue violet), 14(khaki), 15(lime green), 16(gray). And fonts are: 1(variable, courie), 2(12X24, Helv), 3(variable, tms rms), 4(variable, Modern), 5(variable, Script), 6(variable, Roman), 7(variable, System), 8(variable, Preview).

2.5 Cartesian Coordinates and Viewports

The VGLIB5 adapts the right-hand Cartesian coordinate system as both papers and displays that have normally left-hand and physical dot resolution (VGA, SVGA, VGA) coordinate system. User must specify a view port based on the system.

2.6 Device Selection

One integer IDBG(1) at the time of opening is the specific indicator of the target devices. IDBG(1)=IGRPS*1000+IGRVTR*100+IGRPLT*10+IGRCRT, where IGRPS is for PS printer, IGRVTR is for VRT, IGRPLT is for HP-GL Plot, and IGRCRT is for display, respectively. In the VGLIB5, IGRPS =0 (No write for PS), =1 (PS printer), IGRVTR=0 (No output for VTR), =1 (Image Memory --> VTR server), =2 (Image Memory --> IBM/506 --> VTR), =3 (IBM/506-direct --> VTR), IGRPLT =0 (No write for plot tape), =1 (HP-GL Plot), IGRCRT =0 (No display), =1 (PC-CRT), =2 (X-Window), =3 (IBM/M506-direct), =4 (IBM/5080 file), =5 (Image Memory), =6 (Image Memory --> PC-CRT), =7 (Image Memory --> IBM/M506, for IGRVTR=2).

2.7 Calling Sequences

All calling sequences look like the routines of Basic language. Typical examples are:

```
C-series   subroutine  VGCIRCLE2(X,Y,R,ICOLI)
           subroutine  VGCLEAR
           subroutine  VGCLOSE
           subroutine  VGCOLOR4(ICOLI)
           subroutine  VGDISPLAY(IXPOS,IYPOS)
           subroutine  VGFACTOR(XF,YF)
           subroutine  VGFILLA(N,XX,YY,ICOLI)
           subroutine  VGLINE3(X1,Y1,X2,Y2,ICOLI)
           subroutine  VGLINE4(N,XX,YY,ICOLI)
           subroutine  VGNUMBER(X,Y,SIZE,FPN,THETA,M)
           subroutine  VGOPEN(I16,IDBG,CSIZE,PENW)
           subroutine  VGPASIZE(CSIZE)
           subroutine  VGPENTYPE(I,ILIMIT,LNT)
           subroutine  VGPENW(PENW)
           subroutine  VGROTATE(THETA,ITRX,ITRY,ITRXPS,ITRYPS,TRAXCRT,TRAYCRT)
           subroutine  VGREC(ISW)                                      !VTR
           subroutine  VGSTBY(NFRAME)                                  !VTR
           subroutine  VGSIMBOL(X,Y,SIZE,NBCD,THETA,N
           subroutine  VGTEXT(X,Y,HEIGHT,CHARA,THETA,NCHARA,SPACE,ICOLI)
           subroutine  VGUNIT(C)
           subroutine  VGWINDOW(X1,Y1,X2,Y2)
           subroutine  CRTSIZE(IXMAX,IYMAX,IXCRTSIZE,IYCRTSIZE)        !CRT
           subroutine  CRTA4A3(IX0,IY0,IX1,IY1)                        !CRT
           subroutine  CRTXYSET                                        !CRT
           subroutine  PAPERFRAME(XMINCR,YMINCR,XMAXCR,YMAXCR,ICOLI)   !Paper
G-series   subroutine  GCOLOR1(ITEXT,ICOL1,ICOL2)
           subroutine  GCOLOR2(ICOL1,IBGR)
           subroutine  GCOPY(IFUNC)
           subroutine  GCRLOCATE(IX,IY,MODE1)
           subroutine  GCRSOL(MODE1)
           subroutine  GGET(X1,Y1,X2,Y2,IMAGE,LENGTH)
           subroutine  GCRPOS(IX,IY)
           subroutine  GLINE(X1,Y1,X2,Y2,ICOL1,MODE,ICOL2,IFILMASK)
           subroutine  GPAINT(N,XX,YY,ICOL1)
           subroutine  GPAINT2(X,Y,ICOL1,IFILMASK,ICOL2)
           subroutine  GPRESET(X,Y)
           subroutine  GPSET(X,Y,ICOL1)
           INTEGER*4 FUNCTION  GPOINT(X,Y)
           subroutine  GPUT(X1,Y1,X2,Y2,IMAGE,LENGTH,MODE,ICOL1,ICOL2)
           subroutine  GPUTG3(X,Y,NCHAR,CBUF,IDEG10,ICOL1)
           subroutine  GPUTT3(X,Y,NCHAR,CBUF,ICOL1)
           subroutine  GRFLUSH                                         !X-series only
           subroutine  GRFONTS(IFONT,IHEIGHT,IWIDE)
           subroutine  GSVIEW(IX1,IY1,IX2,IY2,NCHAR,CBUF)
           subroutine  GWINDOW(X1,Y1,X2,Y2)
           subroutine  GWTAPE(IWD)
M-series   subroutine  Mouse_LR(KEYSTATE,X,Y)                          !check mouse click
           subroutine  Mouse_M(KEYSTATE,X,Y)                           !check mouse move
           subroutine  Mouse_P(X,Y)                                    !get mouse position
           subroutine  Mouse_R(KEYSTATE,X,Y)                           !check mouse (right)
P-series   subroutine  DASHP(X,Y,W)
           subroutine  NEWPEN(I)
           subroutine  NUMBER(X,Y,SIZE,FPN,THETA,M)
           subroutine  PLOT(X,Y,I)
           subroutine  PLOTS(CBUF,N,LN,I16,IDBG)
           subroutine  PLTAPE(LN)
           subroutine  SYMBOL(X,Y,SIZE,NBCD,THETA,N)
PS-series  subroutine  PSFILLA(N,XX,YY,ICOLI)
           subroutine  PSPENW(PENW)
           subroutine  PSPLOT(X,Y,IPEN)
```

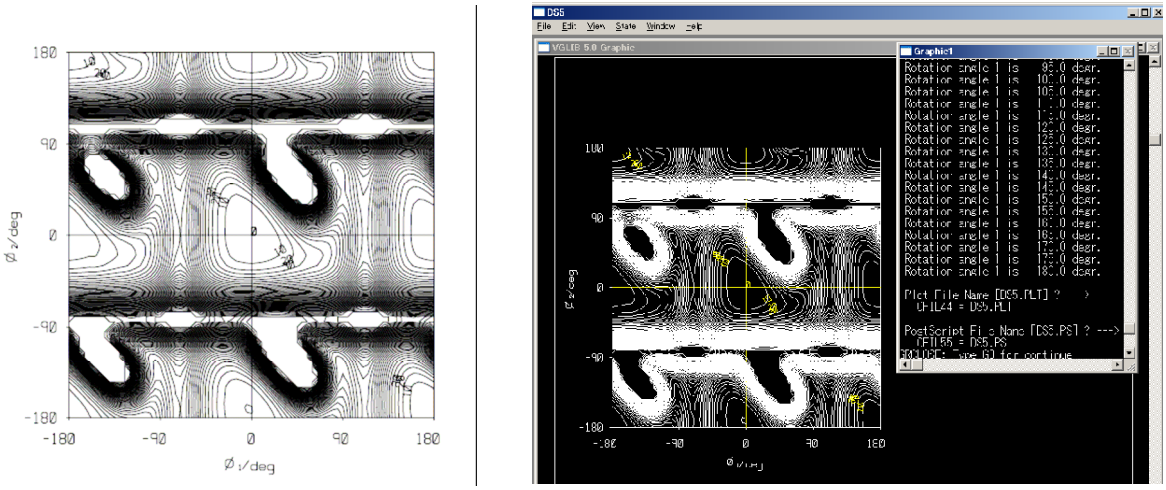
3 Application

The VGLIB5 is used for all graphical subroutines of DS5 [17]. This DS5 (Direct-Searcher automatic system version 5) is an integrated crystallographic program for the crystal structure analysis of organic compounds running on PCs. Each subprogram (ORTEP3, PLUTO, ROTEN, ROTENP, Shake, etc.) in the *DS5* is chosen out eighteen programs from the DS*SYSTEM4 [18]. The ROTEN calculates the difference molecular potential energy using a function of the type $E_{ij} = -A_{ij} r_{ij}^{-6} + B_{ij} \exp(-C_{ij} r_{ij})$ after rotations of one or two molecular fragments about given direction. And the *ROTENP* draws calculated difference potential-energy profiles (1*D*) of the ΔE and contour lines (2*D*) for the rotation of two molecular fragments. Figure 2 shows the subroutine ROTENP source code. Several definitions and calculation parts are omitted for easy understanding.

```
C*============================================================================*
      SUBROUTINE ROTENP(ISWRP,CAT1,DCONT,PRSCALE,PRFACT)                ROTENP
      ...                                                               ROTENP
C***  Calculation                                                       ROTENP
      ...                                                               ROTENP
C***  QuickWin Windows='VGLIB 5.0 Graphic'                              ROTENP
      IDBG(1)=1011       !l.u. is 55=PS, 44=PLT, 33=PC-CRT              ROTENP
                                                                        ROTENP
C***  Open VGLIB5.lib                                                   ROTENP
      CALL VGOPEN('A4R',5.0)                          !PENW=5.0         ROTENP
                                                                        ROTENP
C***  Set CRT Window                                                    ROTENP
C**   Get CRT Size, Paper A4R-size                                      ROTENP
      CALL CRTSIZE(MAXPX,MAXPY,IXCRTSIZE,IYCRTSIZE)  !get current CRT   ROTENP
      CALL CRTA4A3(MX0,MY0,MX1,MY1)                  !get paper MinMax  ROTENP
      CALL GSVIEW(MX0,MY0,MX1,MY1,5,'VGLIB')         !set View      dot ROTENP
      CALL CRTXYSET                                  !get XMINCRT,.. cm ROTENP
      CALL GWINDOW(XMINCRT,YMINCRT,XMAXCRT,YMAXCRT)  !cm, Window, cm    ROTENP
                                                                        ROTENP
C**   Set Color code and Draw Paper Frame                               ROTENP
      ICOLI=7                                                           ROTENP
      CALL VGCOLOR4(ICOLI)                                              ROTENP
      CALL PAPERFRAME(XMINCRT,YMINCRT,XMAXCRT,YMAXCRT,ICOLI)            ROTENP
                                                                        ROTENP
C***  Display One A4R Image                                             ROTENP
      CALL MOJIWK_ROTP                                !X,Y axes         ROTENP
                                                                        ROTENP
C***  Display Profile or Contour lines                                  ROTENP
      IF(NFR.EQ.1) THEN                               !1D profile       ROTENP
        CALL GRAPH_ROTP                                                 ROTENP
      ELSE                                            !2D contour       ROTENP
        CALL CONTR_ROTP(1,361,TH1,1,LOOP1,1,361,TH2,1,LOOP2,            ROTENP
     *                  ENRGP,DCONT,ICOLI)                             ROTENP
      ENDIF                                                             ROTENP
                                                                        ROTENP
C***  Close PS, Plot and CRT                                            ROTENP
      CALL VGCLOSE                                    !Graphics         ROTENP
      RETURN                                                            ROTENP
      END                                                               ROTENP
```

Fig 2: *The source code of ROTENP subprogram*

The HP-GL/Postscript files and display image on PCs from the ROTENP are shown in Fig. 3.



(a) HP-GL/Postscript files (b) Display on PCs

Fig 3: *The output from ROTENP subprogram in DS5.*

Figures 4 and 5 are the output of subroutine Shake, ORTEP3 and PLUTO in the DS5. The Shake (former ShakePSD [19]) is an automatic general phase solver being able to handle both heavy-atom methods and direct methods within one program, and finds out with certainty molecule fragments by a single computer execution. This Shake checks for a heavy atom in the input chemical formula. If the molecule includes a sulphur or a heavier atom, heavy-atom methods are processed to solve the Patterson function (*PSL*) and search for all remaining atoms (*SEARCHER*). Otherwise, the Shake executes in direct methods, calculates normalized structure factors $E_o(\mathbf{h})$ (*NORMAL*) and refines the phase angles $\phi(\mathbf{h})$ using the tangent formula (*TANGENT*) or the minimal function (*MINIMAL*). The outputs of the Shake are 2*D* projection diagrams of the molecules in the unit cell with the final *R*-factor and the bond distances

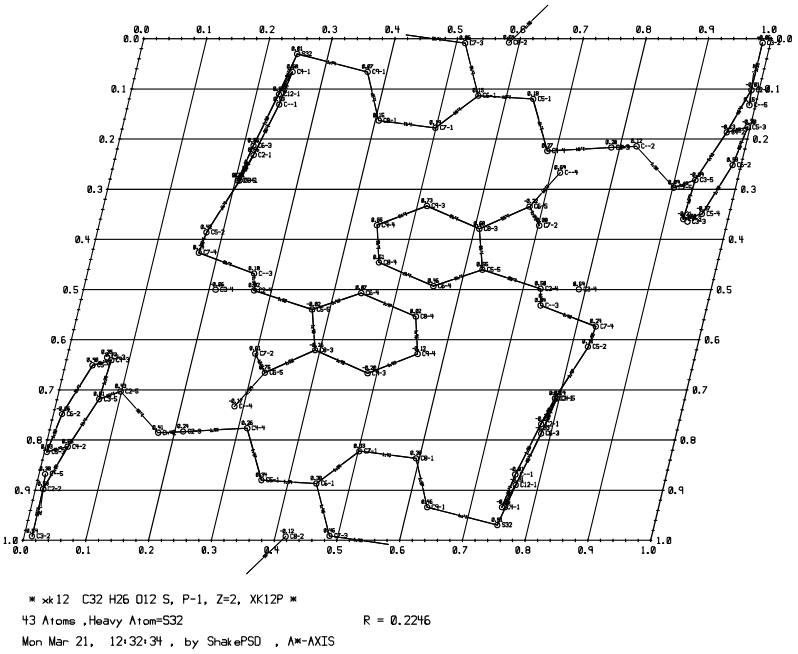


Fig 4: *The output from Shake subprogram in DS5.*

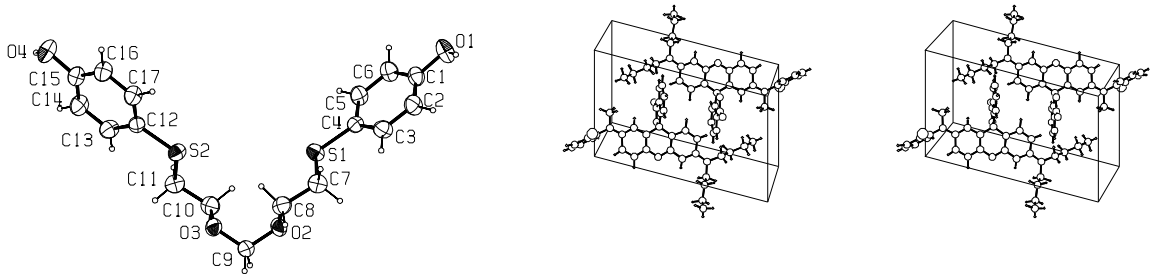


Fig 5: *The output from ORTEP3 and PLUTO subprograms in DS5.*

Another examples shown in Figs. 6 and 7 are the use of mouse in interactive execution of PreLALS workbench [20] for helical polymers. The PreLALS workbench is designed to aid users in the task of visualizing chemical structures using three-dimensional models, editing and the data manipulation for WinLALS program [21] that refines the internal coordinates based on linked-atom least-squares (LALS) method for helical polymers. It combines the views of model building (MB), analyses, and data preparation tools with easy-to-use graphical user interfaces. User can specify an already-defined atom to which it is attached, denoted its precursor, and giving its position in terms of its relation to its precursor in the polymer chain by clicking the displayed atom position. Eight green atoms are the polymer tree sequence that is clicked in order. If user clicks a certain range (ix=0-33 and iy=0-20 dots), the program displays EXIT characters and returns to main program. When other ranges are clicked, the program requests the Re-entry click once more.

```
C*************************************************************************************
      SUBROUTINE MB_Tree(Lchecked)                                              MB_Tree
      ...                                                                       MB_Tree
C                                                                               MB_Tree
      CHARACTER  CATMKIND*1                                                     MB_Tree
      LOGICAL    Lchecked                                                       MB_Tree
      ...                                                                       MB_Tree
C                                                                               MB_Tree
C***  Exit or Retry                                                             MB_Tree
200   CALL Filled_Rectangle(0,0,123,22,6,0)        !6=yellow                    MB_Tree
      CALL GTEXTOUT(1,1,'EXIT or RETRY ?',0,4)     !0=blk,4=couri,h20w16        MB_Tree
      CALL Mouse_LR(KEYSTATE,IXP,IYP)                                           MB_Tree
C                                                                               MB_Tree
C**   EXIT                                                                      MB_Tree
      IF(IXP.LT.33.AND.IYP.LT.20) THEN                                          MB_Tree
        ITRY=0                                                                  MB_Tree
        CALL Clear_Graphic                         !clear display               MB_Tree
        CALL XDraw_Mol                             !display molecule            MB_Tree
        ...                                                                     MB_Tree
        RETURN                                                                  MB_Tree
C                                                                               MB_Tree
C**   Retry                                                                     MB_Tree
      ELSEIF(IXP.GT.65.AND.IXP.LT.105.AND.IYP.LT.15) THEN                       MB_Tree
        ITRY=1                                                                  MB_Tree
        CALL Filled_Rectangle(0,0,35,22,6,0)       !6=yellow                    MB_Tree
        CALL GTEXTOUT(1,1,'EXIT',0,4)              !0=blk,4=couri,h20w16        MB_Tree
        CALL Tree                                  !display tree                MB_Tree
      ENDIF                                                                     MB_Tree
      GO TO 200                                                                 MB_Tree
      END                                                                       MB_Tree
```

Fig 6: *Mouse clicks the tree atoms in PreLALS.*

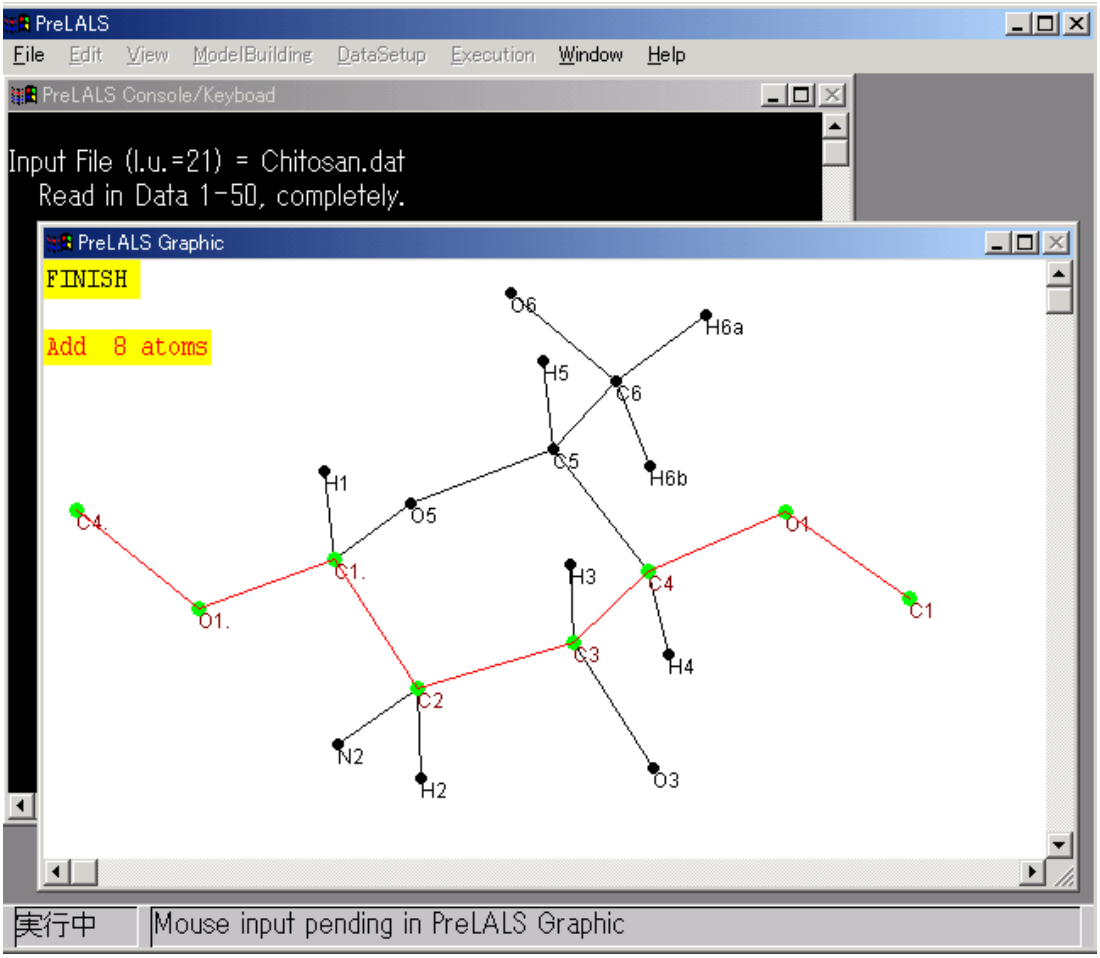


Fig 7: *The tree output from Figure routine of PreLALS.*

4 Conclusion

It is very important for the chemists or researchers that the graphic library which is able to display and draw the graphical results of molecular structure easily exists. If a developer can develop a new graphic software without the dependency on hardware and OS, it is very desirable since the futility of time and the restriction on use will be eased. New developed visual graphic library VGLIB5 can help the software developer who want to use Fortran and/or C languages, to display the scientific results on graphic terminals, to prepare drawing files for HP-GL plotters or postscript files, and to prepare the VTR tape of scientific animation.

References

[1] CCSL, http://www.ill.fr/dif/ccsl/html/ccsldoc.html; CCTBX, http://cci.lbl.gov/; CrysFML, http://www-llb.cea.fr/fullweb/powder.htm; Clipper C++, http://www.yorvic.york.ac.uk/~cowtan/

[2] CRYSTALS and Cameron in C++ with wxWindows, http://www.xtl.ox.ac.uk/; Jana2000 in Fortran 95, http://www-xray.fzu.cz/jana/jana.html; WinGX in Clearwin, http://www.chem.gla.ac.uk/~louis/software/wingx/, OpenGL in C/C++, http://www.ccp14.ac.uk/ccp/web-mirrors/marchingcubefourierviewer/; Maud in Java, http://www.ing.unitn.it/~luttero/; Java applets, http://www.univ-lemans.fr/enseignements/physique/02/; Tcl/Tk in Fortran, http://www.ncnr.nist.gov/xtal; ISAW in Java, http://www.pns.anl.gov/computing/isaw/; CCP4i, http://www.ccp4.ac.uk/martyn/martyn.html; Others (Qt, FOX, etc.), http://www.atai.org/guitool/

[3] ACA-SIGGRAPH, http://www.siggraph.org/

[4] GKS, GKS-3D, a) International Organization for Standardization. The Graphical Kernel System (GKS). Technical Report ISO 7942, ISO Geneva, 1985. b) International Organization for Standardization. The Graphical Kernel System for Three Dimensions (GKS-3D). Technical Report ISO 8805, ISO Geneva, 1987. c) International Organization for Standardization. The Graphical Kernel System (GKS) language bindings -- Part 1: Fortran. Technical Report ISO 8651-1, ISO Geneva, 1988. d) International Organization for Standardization. The Graphical Kernel System for Three Dimensions (GKS-3D) language bindings -- Part 1: Fortran. Technical Report ISO 8806-1, ISO Geneva, 1988.

[5] PHIGS, PEX, a) Getting Started with SunPHIGS- PHIGS Overview, Sun related notes, October 4, 1991. b) http://www.itl.nist.gov/iaui/vvrg/cugini/pvt/hy-std.html c) http://www.faqs.org/faqs/graphics/pex-faq/

[6] Postscript, http://www.adobe.com/products/postscript/main.html

[7] X-Windows, http://www.x.org/

[8] OpenGL http://www.opengl.org/ and http://www.sgi.com/products/software/opengl/

[9] PGPLOT, http://www.astro.caltech.edu/~tjp/pgplot/

[10] Gnuplot, http://www.gnuplot.info/

[11] Alias, http://www.alias.com/eng/index.shtml

[12] AVS, http://www.avs.com/index_wf.html

[13] Okada, K. & Koyama, H., (1991). J. Appl. Cryst., **24(6)**, 1067-1070.

[14] Okada, S. & Okada, K. (1996). Comput. Chem., **20(2)**, 267-270.

[15] Okada, K. & Okada, S. (1997). J. Chem. Inf. Comput. Sci., **37(3)**, 522-528.

[16] Okada, S. & Okada, K. (2000). Z. Kristallogr., **215**, 131-143.

[17] Okada, K. & Boochathum, P. (2005). J. Appl. Cryst., **38**, 842-846.

[18] Okada, S. & Okada, K., (2000). Z. Kristallogr., **215**, 131-143.

[19] Okada, S. & Okada, K., (2000). J. Appl. Cryst., **33(2)**, 406-414.

[20] Okada, K., Boochathum, P., Noguchi, K., Okuyama, K. & Katsube, Y. (2005) J. Comput. Aided Chem., **6**, 12-22.

[21] Okada, K., Noguchi, K., Okuyama, K. & Arnott, S., (2003). Comput. Biol. Chem., **27(3)**, 265-285.

Notes on the calculation of the derivatives for least-squares crystal structure refinement

(updated 6th June 2008)

Riccardo Spagna

Istituto di Cristallografia - CNR, Sede di Monterotondo, Area della Ricerca di Roma 1, Via Salaria Km. 29, 00016 Monterotondo Stazione (Roma), Italy, E-mail: riccardo.spagna@ic.cnr.it ; WWW: http://www.ic.cnr.it/spagna.php

I. Introduction

This is not an original paper on least squares methods. This is only a contribution to the crystallographic community on well established algorithms to calculate derivatives useful for crystal structure refinement using the least squares methods. Almost all of the equations collected here can be found in the literature. The aim of this work is to help the next generation of crystallographers who wants to face with computing problems by least squares methods: they can find here many of the things they need.

All the algorithms described here are implemented in the SIR package (Burla, *et al.*, 2005)

II. Background

The least-squares method is the refinement method most used for small-medium size molecules. In the following, we recall briefly the procedure commonly used (see also Giacovazzo, 2002).

The structure factor may be written as:

$$F_h = \sum_j f_j \exp\left(2\pi i \cdot \mathbf{h} \cdot \mathbf{r_j}\right)$$

where f_j is the scattering factor of the *j-th* atom in the unit cell at the $\sin\theta_h$ value of **h** (=*hkl*), including the exponential term for the atomic thermal motion; $\mathbf{r_j}$ are the positional parameters (*x,y,z*) of the *j-th* atom.
To find the best atomic parameters describing the structure, the function being minimized by the least-squares method is the residual:

$$M = \sum_h w_h \left(|F_h|^2_{oss} - K^2 |F_h|^2_{calc}\right)^2 = \sum_h w_h \left(\Delta F^2_h\right)^2$$

where $|F_h|^2_{oss}$ is the observed intensity and w_h is weights to be assigned to each of them.
The theory of least squares suggests for w_h the relationships:

$$w_h \approx \frac{1}{\sigma_h^2}$$

Many authors have studied the problem concerning the choice of the weights and considered functions based on the observations or on a mix of these with the estimated standard uncertainty σ_h. Examples of these weighting schemes were suggested by Cruickshank (1965), *i.e.*

$$w_h = 1 \Big/ \left(a + b|F_h| + c|F_h|^2\right) ,$$

and by Sheldrick, used in the program SHELX:

$$w_h = 1 \Big/ \left(\sigma_h^{\,2} + bP + cP^2 \right) ,$$

where

$$P = \left(F_{oss}^2 + 2F_{calc}^2 \right) \Big/ 3$$

See also Spagna & Camalli, (1999) for an extensive analysis of weighting schemes.

Because the $|F_h|^2_{calc}$ values are not linear functions of the m atomic parameters r_j, approximate values are obtained by expanding in a Taylor series around r_j, and truncating after the first term:

$$M = \sum_h w_h \left(|F_h|^2_{oss} - K^2 \left\{ |F_h|^2_{calc} + \sum_j 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right\} \right)^2 =$$

$$= \sum_h w_h \left(\Delta F^2{}_h - \sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right)^2$$

M is a minimum with respect to the Δr_i if the derivates $\partial M / \partial \Delta r_i$ are zero:

$$\frac{\partial M}{\partial \Delta r_i} = \sum_h w_h \left(\Delta F^2{}_h - \sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right) \left(K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i} \right) = 0 \qquad \text{for } i=1,\ldots,m$$

and the normal equations are obtained:

$$\sum_h w_h \left(\sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \right) \left(K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i} \Delta r_i \right) = \sum_h w_h \Delta F^2{}_h K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i}$$

Solving these equations, the shifts Δr_i are obtained and applied to the parameters r_i . Several cycles of refinement are required to achieve convergence.

Statistical descriptors (Schwarzenbach et al, 1989) have to be considered for evaluating the refinement, as the classical R factor

$$R = \frac{\sum_h \left| |F_h|_{oss} - \left(K|F_h| \right)_{calc} \right|}{\sum_h |F_h|_{oss}} ,$$

the weighted wR factor

$$wR = \sqrt{\frac{\sum_h w_h \left(|F_h|^2_{oss} - \left(K|F_h| \right)^2_{calc} \right)^2}{\sum_h w_h \left(|F_h|^2_{oss} \right)^2}} ,$$

the goodness of fit (where n-m is the number of degrees of freedom, n = number of observations and m = number of variables of the model)

$$S = \sqrt{\frac{\sum_h w_h \left(|F_h|^2_{oss} - \left(K|F_h| \right)^2_{calc} \right)^2}{(n-m)}}$$

In some case (for example when the data are bad or not enough), we can introduce some restraints (this technique increases the number of observations). The function being minimized becomes:

$$M = \sum_h w_h \left(|F_h|^2_{oss} - K^2 |F_h|^2_{calc} \right)^2 + \sum_q w_q \left(g_q - g_{qo} \right)^2 ,$$

where g_{qo} is the function of the q-th restraint, g_q is its optimal value and w_q is the weight.
The approximate values are obtained expanding the g_q's in Taylor series.

$$M = \sum_h w_h \left(|F_h|^2_{oss} - K^2 \left\{ |F_h|^2_{calc} + \sum_j 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right\} \right)^2 + \sum_q w_q \left(g_q - g_{qo} - \sum_j \frac{\partial g_{qo}}{\partial r_j} \Delta r_j \right)^2 =$$

$$\sum_h w_h \left(\Delta F^2{}_h - \sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right)^2 + \sum_q w_q \left(\Delta g - \sum_j \frac{\partial g_{qo}}{\partial r_j} \Delta r_j \right)^2$$

Setting to zero the derivates of M with respect to the Δr_i ,

$$\sum_h w_h \left(\Delta F^2{}_h - \sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \Delta r_j \right) \left(K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i} \right) + \sum_q w_q \left(\Delta g - \sum_j \frac{\partial g_{qo}}{\partial r_j} \Delta r_j \right) \frac{\partial g_{qo}}{\partial r_i} = 0$$

the normal equations are obtained:

$$\left\{ \sum_h w_h \left(\sum_j K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_j} \right) \left(K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i} \right) + \sum_q w_q \left(\sum_j \frac{\partial g_{qo}}{\partial r_j} \right) \frac{\partial g_{qo}}{\partial r_i} \right\} \Delta r_i =$$

$$= \sum_h w_h \Delta F^2{}_h K^2 2|F_h|_{calc} \frac{\partial |F_h|_{calc}}{\partial r_i} + \sum_q w_q \Delta g \frac{\partial g_{qo}}{\partial r_i}$$

The system of normal equations can be written

$\mathbf{M}\Delta\mathbf{r} = \mathbf{b}$,

where **M** is the n^2 symmetric normal matrix, $\Delta\mathbf{r}$ is the unknown shift and **b** is the know term**.** $\Delta\mathbf{r}$ and **b** are vectors of order n. The solution $\Delta\mathbf{r}$ is obtained by inverting the normal matrix and by multiplying it by the **b** vector:

$\Delta\mathbf{r} = \mathbf{b}\mathbf{M}^{-1}$

The diagonal terms $M^{-1}{}_{ii}$ provide an approximate value of the standard uncertainties of the variables r_i: such variables r_i can be normalized by dividing them by the quantity $M_{ii}^{1/2}$.

The corresponding normalized matrix is obtained by replacing the M_{ij} elements by:

$$M'_{ij} = \frac{M_{ij}}{\sqrt{M_{ii} M_{jj}}}$$

The normalized M'_{ij} value corresponds to the correlation coefficient between the parameters r_i and r_j and values less than unity are normally obtained. Values greater than unity show evidence of strong correlation between the i and j parameters.

III. Structure factor derivatives

For a crystal structure with s symmetry operators of the space group and m symmetry-independent atoms, a generalized form of the structure factor (the subscript *calc* is omitted) is:

1) for isotropic thermal motion:

$$F_h = \sum_{n=1,s} \left(\sum_{i=1,m} f_i \exp(2\pi i \cdot \mathbf{h} \cdot \mathbf{r}_i) \exp\left[-\frac{\mathrm{B}_i}{4} \left(\frac{2\sin\vartheta_h}{\lambda} \right)^2 \right] (occ_i) \right),$$

2) for anisotropic thermal motion:

$$F_h = \sum_{n=1,s} \left(\sum_{i=1,m} f_i \exp(2\pi i \cdot \mathbf{h} \cdot \mathbf{r}_i) \exp\left[-\sum_i \sum_j \beta_{ij} h_i h_j \right] (occ_i) \right)$$

The first sum is over the *s* symmetry operators and the inner sum is over the *m* atoms; occ_i is the site occupation of *i-th* atom (normally = 1.0, different values are related to possible vacancies in the crystal, or to special positions of the atom);

$$B_i = 8\pi^2 \left\langle u_i^{\,2} \right\rangle ,$$

is the isotropic T.F (thermal factor) (where $<u_i^2>$ is the mean squares amplitude of the thermal vibration of the atom).

Writing the reciprocal of the interplanar spacing d_h:

$$\left(\frac{2\sin\vartheta_h}{\lambda}\right)^2 = \left(\frac{1}{d_h}\right)^2 = h^2a^{*2} + k^2b^{*2} + l^2c^{*2} + 2hka^*b^*\cos\gamma^* + 2hla^*c^*\cos\beta^* + 2klb^*c^*\cos\alpha^* \text{ we}$$

can represent the T.F. in the general temperature factor expression

$$T.F. = \exp\left[-\left(\beta_{11}h^2 + \beta_{22}k^2 + \beta_{33}l^2 + \beta_{12}hk + \beta_{13}hl + \beta_{23}kl\right)\right] ,$$

where β_{ij} are the six independent components of the anisotropic thermal motion described as an ellipsoid.

The anisotropic thermal parameter can be written also as function on U_{ij} (parameters expressed in terms of mean-squares amplitudes of vibration in Å) or on B_{ij} (in the same units as the conventional isotropic thermal parameter B):

$$T.F. = \exp\left[-2\pi^2 \sum_i \sum_j U_{ij} h_i h_j \mathbf{a}_i^* \mathbf{a}_j^*\right],$$

$$T.F. = \exp\left[-\frac{1}{4} \sum_i \sum_j B_{ij} h_i h_j \mathbf{a}_i^* \mathbf{a}_j^*\right]$$

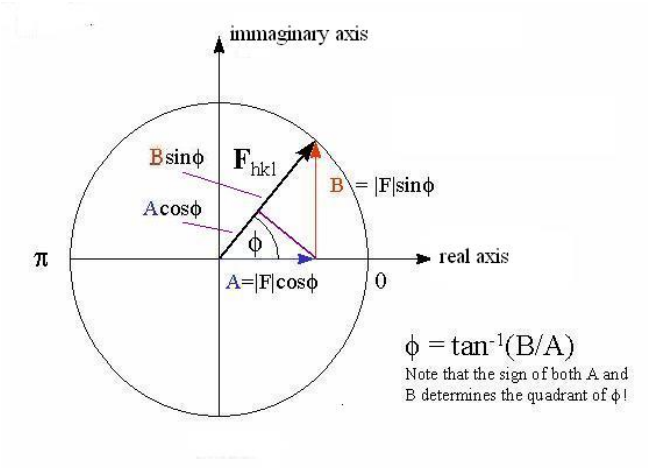
In the following, we consider the β_{ij} parameters.

The F_h. may be written as:

$$F_h = A_h + iB_h = |F_h|\exp(i\phi)$$

$$|F_h| = A_h\cos\phi + B_h\sin\phi \qquad (3,1)$$

$$|F_h| = \sqrt{A_h^{\,2} + B_h^{\,2}} \qquad (3,2)$$

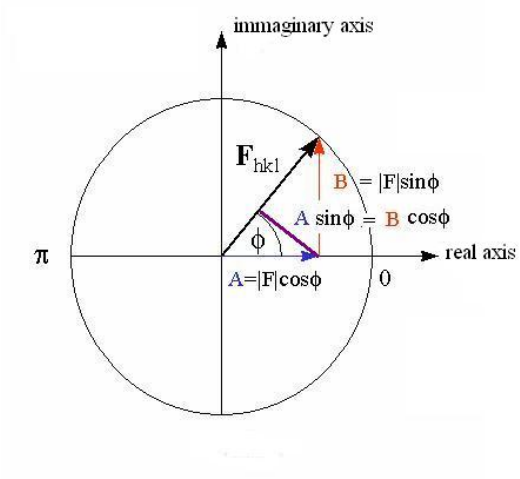


Hence, if p_j is a parameter of the *j-th* atom, deriving the equation (3,1), gives

$$\frac{\partial|F_h|}{\partial p_j} = \frac{\partial A_h}{\partial p_j}\cos\phi + \frac{\partial B_h}{\partial p_j}\sin\phi + \left(-A_h\sin\phi + B_h\cos\phi\right)\frac{\partial\phi}{\partial p_j}$$

It is easy to see that

$$\left(-A_h\sin\phi + B_h\cos\phi\right) = 0$$



Thus,

$$\frac{\partial|F_h|}{\partial p_j}=\frac{\partial A_h}{\partial p_j}\cos\phi+\frac{\partial B_h}{\partial p_j}\sin\phi \qquad (3,3)$$

Of course, we can arrive to the same result by deriving the equation (3,2). We obtain

$$\frac{\partial|F_h|}{\partial p_j}=\frac{1}{2|F_h|}\left(2A_h\frac{\partial A_h}{\partial p_j}+2B_h\frac{\partial B_h}{\partial p_j}\right)=\frac{2A_h}{2|F_h|}\frac{\partial A_h}{\partial p_j}+\frac{2B_h}{2|F_h|}\frac{\partial B_h}{\partial p_j}=\frac{A_h}{|F_h|}\frac{\partial A_h}{\partial p_j}+\frac{B_h}{|F_h|}\frac{\partial B_h}{\partial p_j}$$

Replacing

$$\frac{A_h}{|F_h|}=\cos\phi$$

$$\frac{B_h}{|F_h|}=\sin\phi$$

gives the previous relationship (3,3).

Accordingly

$$\frac{\partial|F_h|^2}{\partial p_j}=2|F_h|\frac{\partial|F_h|}{\partial p_j}=2|F_h|\left(\frac{\partial A_h}{\partial p_j}\cos\phi+\frac{\partial B_h}{\partial p_j}\sin\phi\right)=2A_h\frac{\partial A_h}{\partial p_j}+2B_h\frac{\partial B_h}{\partial p_j} \qquad (3,4)$$

We may obtain (3,4) directly from the relationship:

$$|F_h|^2=\left(A_h^{\,2}+B_h^{\,2}\right)$$

We calculate the explicit expressions of the F_h derivatives with respect to the unknown variables x_j, y_j, z_j, B_j, $\beta_{(11)j}$ $\beta_{(12)j}$ $\beta_{(13)j}$.., occ_j and overall parameters K and $B(overall)$, considering isotropic T.F. and for sake of simplicity omitting the explicit indication of the loop on symmetry operators

$$A_h=\sum_j f_j\cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right)=\sum_j A_j$$

$$B_h=\sum_j f_j\sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right)=\sum_j B_j$$

Atomic coordinates x_j :

$$\frac{\partial A_j}{\partial x_j}=\left(-2\pi h\right)f_j\sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right)=-2\pi hB_j$$

$$\frac{\partial B_j}{\partial x_j}=\left(2\pi h\right)f_j\cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right)=2\pi hA_j$$

$$\frac{\partial |F_h|}{\partial x_j} = \frac{\partial A_h}{\partial x_j}\cos\phi + \frac{\partial B_h}{\partial x_j}\sin\phi = \frac{A_h}{|F_h|}\frac{\partial A_h}{\partial x_j} + \frac{B_h}{|F_h|}\frac{\partial B_h}{\partial x_j} = \frac{A_h}{|F_h|}\frac{\partial A_j}{\partial x_j} + \frac{B_h}{|F_h|}\frac{\partial B_j}{\partial x_j} = \frac{A_h}{|F_h|}\left(-2\pi hB_j\right) + \frac{B_h}{|F_h|}\left(2\pi hA_j\right)$$

$$\frac{\partial A_j}{\partial y_j} = \left(-2\pi k\right)f_j \sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = -2\pi kB_j$$

$$\frac{\partial B_j}{\partial y_j} = \left(2\pi k\right)f_j \cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = 2\pi kA_j$$

$$\frac{\partial |F_h|}{\partial y_j} = \frac{\partial A_h}{\partial y_j}\cos\phi + \frac{\partial B_h}{\partial y_j}\sin\phi = \frac{A_h}{|F_h|}\left(-2\pi kB_j\right) + \frac{B_h}{|F_h|}\left(2\pi kA_j\right)$$

$$\frac{\partial A_j}{\partial z_j} = \left(-2\pi l\right)f_j \sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = -2\pi lB_j$$

$$\frac{\partial B_j}{\partial z_j} = \left(2\pi l\right)f_j \cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = 2\pi lA_j$$

$$\frac{\partial |F_h|}{\partial z_j} = \frac{\partial A_h}{\partial z_j}\cos\phi + \frac{\partial B_h}{\partial z_j}\sin\phi = \frac{A_h}{|F_h|}\left(-2\pi lB_j\right) + \frac{B_h}{|F_h|}\left(2\pi lA_j\right)$$

Atomic isotropic thermal parameters B_j :

$$\frac{\partial A_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 f_j \cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 A_j$$

$$\frac{\partial B_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 f_j \sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right]\left(occ_j\right) = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 B_j$$

$$\frac{\partial |F_h|}{\partial \mathrm{B_j}} = \frac{\partial A_h}{\partial \mathrm{B_j}}\cos\phi + \frac{\partial B_h}{\partial \mathrm{B_j}}\sin\phi = \frac{A_h}{|F_h|}\left[-\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 A_j\right] + \frac{B_h}{|F_h|}\left[-\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 B_j\right]$$

Atomic anisotropic thermal parameters β_{ij} :

$$\frac{\partial A_j}{\partial \beta_{(11)_j}} = \left(-h^2\right)f_j \cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathbf{h}\beta_{ij}\mathbf{h}\right]\left(occ_j\right) = \left(-h^2\right)A_j$$

$$\frac{\partial B_j}{\partial \beta_{(11)_j}} = \left(-h^2\right)f_j \sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathbf{h}\beta_{ij}\mathbf{h}\right]\left(occ_j\right) = \left(-h^2\right)B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(11)_j}} = \frac{\partial A_h}{\partial \beta_{(11)_j}}\cos\phi + \frac{\partial B_h}{\partial \beta_{(11)_j}}\sin\phi = \frac{A_h}{|F_h|}\left[\left(-h^2\right)A_j\right] + \frac{B_h}{|F_h|}\left[\left(-h^2\right)B_j\right]$$

$$\frac{\partial A_j}{\partial \beta_{(12)_j}} = \left(-hk\right)f_j \cos\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[-\mathbf{h}\beta_{ij}\mathbf{h}\right]\left(occ_j\right) = \left(-hk\right)A_j$$

$$\frac{\partial B_j}{\partial \beta_{(12)_j}} = \left(-hk\right)f_j \sin\left(2\pi\cdot\mathbf{h}\cdot\mathbf{r_j}\right)\exp\left[\mathbf{h}\beta_{ij}\mathbf{h}\right]\left(occ_j\right) = \left(-hk\right)B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(12)_j}} = \frac{\partial A_h}{\partial \beta_{(12)_j}}\cos\phi + \frac{\partial B_h}{\partial \beta_{(12)_j}}\sin\phi = \frac{A_h}{|F_h|}\left[\left(-hk\right)A_j\right] + \frac{B_h}{|F_h|}\left[\left(-hk\right)B_j\right]$$

$$\frac{\partial A_j}{\partial \beta_{(13)_j}} = (-hl) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-hl) A_j$$

$$\frac{\partial B_j}{\partial \beta_{(13)_j}} = (-hl) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-hl) B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(13)_j}} = \frac{\partial A_h}{\partial \beta_{(13)_j}} \cos\phi + \frac{\partial B_h}{\partial \beta_{(13)_j}} \sin\phi = \frac{A_h}{|F_h|}[(-hl) A_j] + \frac{B_h}{|F_h|}[(-hl) B_j]$$

$$\frac{\partial A_j}{\partial \beta_{(22)_j}} = (-k^2) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-k^2) A_j$$

$$\frac{\partial B_j}{\partial \beta_{(22)_j}} = (-k^2) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-k^2) B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(22)_j}} = \frac{\partial A_h}{\partial \beta_{(22)_j}} \cos\phi + \frac{\partial B_h}{\partial \beta_{(22)_j}} \sin\phi = \frac{A_h}{|F_h|}[(-k^2) A_j] + \frac{B_h}{|F_h|}[(-k^2) B_j]$$

$$\frac{\partial A_j}{\partial \beta_{(23)_j}} = (-kl) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-kl) A_j$$

$$\frac{\partial B_j}{\partial \beta_{(23)_j}} = (-kl) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-kl) B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(23)_j}} = \frac{\partial A_h}{\partial \beta_{(23)_j}} \cos\phi + \frac{\partial B_h}{\partial \beta_{(23)_j}} \sin\phi = \frac{A_h}{|F_h|}[(-kl) A_j] + \frac{B_h}{|F_h|}[(-kl) B_j]$$

$$\frac{\partial A_j}{\partial \beta_{(33)_j}} = (-l^2) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-l^2) A_j$$

$$\frac{\partial B_j}{\partial \beta_{(33)_j}} = (-l^2) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}](occ_j) = (-l^2) B_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(33)_j}} = \frac{\partial A_h}{\partial \beta_{(33)_j}} \cos\phi + \frac{\partial B_h}{\partial \beta_{(33)_j}} \sin\phi = \frac{A_h}{|F_h|}[(-l^2) A_j] + \frac{B_h}{|F_h|}[(-l^2) B_j]$$

Site occupation factors occ_j:

$$\frac{\partial A_j}{\partial occ_j} = \frac{A_j}{occ_j}$$

$$\frac{\partial B_j}{\partial occ_j} = \frac{B_j}{occ_j}$$

$$\frac{\partial |F_h|}{\partial occ_j} = \frac{\partial A_h}{\partial occ_j} \cos\phi + \frac{\partial B_h}{\partial occ_j} \sin\phi = \frac{A_h}{|F_h|} \frac{A_j}{occ_j} + \frac{B_h}{|F_h|} \frac{B_j}{occ_j}$$

Overall isotropic vibrations *B(overall)*:

$$\frac{\partial |F_h|}{\partial B(overall)} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 |F_h|$$

All the previous derivatives have to be multiplied by ($w_h K^2$), where K is the factor to put F_{calc} on the same scale as F_{obs} , and w_h are the weights.

Overall scale factor K:

$$w_h \frac{\partial |F_h|}{\partial K} = w_h |F_h|$$

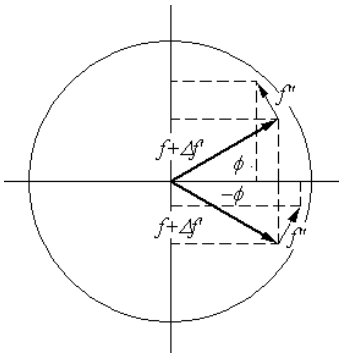
and

$$w_h \frac{\partial K^2 |F_h|^2}{\partial K} = w_h 2K |F_h|^2$$

IV. Effect of anomalous scattering

Under particular conditions, the atomic scattering factor f_0, value defined away from the absorption edge, is modified by introducing two quantities, one real $\Delta f'$ (with the same direction of f_0) and one imaginary $\Delta f''$ with a phase angle of 90° respect to the real component, called real and imaginary dispersion correction respectively. In these case the scattering is called anomalous:

$$f = f_0 + \Delta f' + i\Delta f''$$



Therefore, the real correction is added to the normal scattering factor:

$$A_h = \sum_j (f_0 + \Delta f')_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[- \mathrm{B_j} \left(\frac{\sin \vartheta_h}{\lambda} \right)^2 \right] (occ_j) = \sum_j A_j$$

$$B_h = \sum_j (f_0 + \Delta f')_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[- \mathrm{B_j} \left(\frac{\sin \vartheta_h}{\lambda} \right)^2 \right] (occ_j) = \sum_j B_j$$

while for the imaginary correction $i\Delta f''$ we define the structure factor contributions:

$$C_h = \sum_j \Delta f''_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[- \mathrm{B_j} \left(\frac{\sin \vartheta_h}{\lambda} \right)^2 \right] (occ_j) = \sum_j C_j = \sum_j \frac{\Delta f''}{(f_0 + \Delta f')} A_j$$

$$D_h = \sum_j \Delta f''_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[- \mathrm{B_j} \left(\frac{\sin \vartheta_h}{\lambda} \right)^2 \right] (occ_j) = \sum_j D_j = \sum_j \frac{\Delta f''}{(f_0 + \Delta f')} B_j$$

Under this condition, the F_h. may be written as:

$$F_h = (A_h - D_h) + i(B_h + C_h)$$

$$\tan\phi_h = \frac{(B_h + C_h)}{(A_h - D_h)}$$

$$|F_h| = (A_h - D_h)\cos\phi_h + (B_h + C_h)\sin\phi_h = \left(A_h - \sum_j \frac{\Delta f''}{(f_0 + f')} B_j\right)\cos\phi_h + \left(B_h + \sum_j \frac{\Delta f''}{(f_0 + f')} A_j\right)\sin\phi_h$$

$$\frac{\partial|F_h|}{\partial p_j} = \left(\frac{\partial A_h}{\partial p_j} - \frac{\partial D_h}{\partial p_j}\right)\cos\phi + \left(\frac{\partial B_h}{\partial p_j} + \frac{\partial C_h}{\partial p_j}\right)\sin\phi$$

and F_{-h}:

$$F_{-h} = (A_h + D_h) - i(B_h - C_h)$$

$$\tan\phi_{-h} = \frac{(-B_h + C_h)}{(A_h + D_h)}$$

$$|F_{-h}| = (A_h + D_h)\cos\phi_{-h} + (-B_h + C_h)\sin\phi_{-h}$$

The derivatives of F_h and F_{-h} are calculated respect to p_j considering the general equations:

$$\frac{\partial|F_h|}{\partial p_j} = \left(\frac{\partial A_h}{\partial p_j} - \frac{\partial D_h}{\partial p_j}\right)\cos\phi + \left(\frac{\partial B_h}{\partial p_j} + \frac{\partial C_h}{\partial p_j}\right)\sin\phi$$

For **h**: For –**h**:

Atomic coordinates $\mathbf{x}_j$:

$$\frac{\partial A_j}{\partial \mathbf{x}_j} = -2\pi\mathbf{h}B_j \qquad \frac{\partial A_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}B_j$$

$$\frac{\partial B_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}A_j \qquad \frac{\partial B_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}A_j$$

$$\frac{\partial C_j}{\partial \mathbf{x}_j} = \frac{\Delta f''}{(f_0 + \Delta f')}\frac{\partial A_j}{\partial \mathbf{x}_j} = -2\pi\mathbf{h}\frac{\Delta f''}{(f_0 + \Delta f')}B_j = \qquad \frac{\partial C_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}\frac{\Delta f''}{(f_0 + \Delta f')}B_j = 2\pi\mathbf{h}D_j$$

$$= -2\pi\mathbf{h}D_j$$

$$\frac{\partial D_j}{\partial \mathbf{x}_j} = \frac{\Delta f''}{(f_0 + \Delta f')}\frac{\partial B_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}\frac{\Delta f''}{(f_0 + \Delta f')}A_j = \qquad \frac{\partial D_j}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}\frac{\Delta f''}{(f_0 + \Delta f')}A_j = 2\pi\mathbf{h}C_j$$

$$= 2\pi\mathbf{h}C_j$$

$$\frac{\partial|F_h|}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}(-B_j - C_j)\cos\phi_h + 2\pi\mathbf{h}(A_j - D_j)\sin\phi_h$$

$$\frac{\partial|F_{-h}|}{\partial \mathbf{x}_j} = 2\pi\mathbf{h}(B_j - C_j)\cos\phi_{-h} + 2\pi\mathbf{h}(A_j + D_j)\sin\phi_{-h}$$

Atomic isotropic thermal parameters B_j

$$\frac{\partial A_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 A_j \qquad \frac{\partial A_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 A_j$$

$$\frac{\partial B_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 B_j \qquad \frac{\partial B_j}{\partial \mathrm{B_j}} = +\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 B_j$$

$$\frac{\partial C_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 C_j \qquad \frac{\partial C_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 C_j$$

$$\frac{\partial D_j}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 D_j \qquad \frac{\partial D_j}{\partial \mathrm{B_j}} = +\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 D_j$$

$$\frac{\partial |F_h|}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 \left\{\left(A_j - D_j\right)\cos\phi_h + \left(B_j + C_j\right)\sin\phi_h\right\}$$

$$\frac{\partial |F_{-h}|}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda}\right)^2 \left\{\left(A_j + D_j\right)\cos\phi_{-h} + \left(-B_j + C_j\right)\sin\phi_{-h}\right\}$$

Atomic anisotropic thermal parameters β_{ij} :

$$\frac{\partial A_j}{\partial \beta_{(11)j}} = \left(-h^2\right)A_j \qquad \frac{\partial A_j}{\partial \beta_{(11)j}} = \left(-h^2\right)A_j$$

$$\frac{\partial B_j}{\partial \beta_{(11)j}} = \left(-h^2\right)B_j \qquad \frac{\partial B_j}{\partial \beta_{(11)j}} = \left(h^2\right)B_j$$

$$\frac{\partial C_j}{\partial \beta_{(11)\mathrm{j}}} = \left(-h^2\right)C_j \qquad \frac{\partial C_j}{\partial \beta_{(11)\mathrm{j}}} = \left(-h^2\right)C_j$$

$$\frac{\partial D_j}{\partial \beta_{(11)\mathrm{j}}} = \left(-h^2\right)D_j \qquad \frac{\partial D_j}{\partial \beta_{(11)\mathrm{j}}} = \left(h^2\right)D_j$$

$$\frac{\partial |F_h|}{\partial \beta_{(11)j}} = \left(-h^2\right)\left\{\left(A_j - D_j\right)\cos\phi_h + \left(B_j + C_j\right)\sin\phi_h\right\}$$

$$\frac{\partial |F_{-h}|}{\partial \beta_{(11)j}} = \left(-h^2\right)\left\{\left(A_j + D_j\right)\cos\phi_{-h} + \left(-B_j + C_j\right)\sin\phi_{-h}\right\}$$

and so on for the other parameters β_{ij}

Site occupation factors occ_j :

$$\frac{\partial A_j}{\partial occ_j} = \frac{A_j}{occ_j} \qquad \frac{\partial A_j}{\partial occ_j} = \frac{A_j}{occ_j}$$

$$\frac{\partial B_j}{\partial occ_j} = \frac{B_j}{occ_j} \qquad \frac{\partial B_j}{\partial occ_j} = -\frac{B_j}{occ_j}$$

$$\frac{\partial C_j}{\partial occ_j} = \frac{C_j}{occ_j} \qquad \frac{\partial C_j}{\partial occ_j} = \frac{C_j}{occ_j}$$

$$\frac{\partial D_j}{\partial occ_j} = \frac{D_j}{occ_j} \qquad \frac{\partial D_j}{\partial occ_j} = -\frac{D_j}{occ_j}$$

$$\frac{\partial |F_h|}{\partial occ_j} = \frac{1}{occ_j}\left\{\left(A_j - D_j\right)\cos\phi_h + \left(B_j + C_j\right)\sin\phi_h\right\}$$

$$\frac{\partial |F_{-h}|}{\partial occ_j} = \frac{1}{occ_j}\left\{\left(A_j + D_j\right)\cos\phi_{-h} + \left(-B_j + C_j\right)\sin\phi_{-h}\right\}$$

V. Inclusion of secondary extinction correction

The integrated intensity of a diffracted beam in the kinematical approximation is given by:

$I_h = I_0 K_1 K_2 LPTY|F_h|^2$

Where:

I_0 e' l'intensità del fascio incidente

$K_1 = \left(\frac{e^2}{mc^2}\right)^2$ takes into account the universal constants in the Thomson equation ;

$K_2 = \frac{V_x}{V^2}\lambda^3$ (V_x is the volume of the crystal and V is the volume of the unit cell).

$L = \frac{1}{\sin 2\theta}$ is the Lorentz coefficient

P is the polarization correction

T is the absorption correction

$Y = \left[1 + \left(\frac{2p_2}{p_1}\right)X_0\right]^{-1/2}$ is the extinction coefficient (where $p_n = 1 + \cos^{2n} 2\theta$ is the polarization correction).

$|F_h|$ is the structure factor.

We may write the the integrated intensity:

$$I_h = I_0\left(\frac{e^2}{mc^2}\right)^2\left(\frac{V_x}{V^2}\lambda^3\right)\left(\frac{1}{\sin 2\theta}\right)\left(\frac{\cos^2 2\theta_M + \cos^4 2\theta}{\cos^2 2\theta_M + \cos^2 2\theta}\right)T\left[1 + \left(\frac{2p_2}{p_1}\right)X_0\right]^{-1/2}|F_h|^2$$

We report the procedure to refine the extinction parameter following A.C. Larson (1969) .

The structure factor corrected for extinction factor is defined as:

$$|F_c^*| = |F_c|\left(1 + 2r^* Q_0 T\frac{p_2}{p_1}\right)^{-1/4} = |F_c|\left(1 + 2r^*|F_c|^2\delta\right)^{-1/4}$$

Where r^* is a function of the extinction factor and δ is:

$$\delta = \frac{\lambda^3}{V^2}\left(\frac{e^2}{mc^2}\right)^2 \bar{t}\,\frac{1}{\sin 2\theta}\,\frac{\cos^2 2\theta_M + \cos^4 2\theta}{\cos^2 2\theta_M + \cos^2 2\theta}$$

The Thompson differential cross section σ for the charged particles (area/solid angle) is

$$\sigma = \left(\frac{q^2}{mc^2} \right)^2$$

where q is the charge per particle, and m is the mass per particle. Note that this is the square of the classical radius of a point particle of mass m and charge q. For an electron, the differential cross section is then:

$$\sigma = 7.94079...x10^{-26}\ cm^2/sr$$

and set $\bar{t} = 1.0$ ($\bar{t}$ takes into account of the transmission factor and of the V_x volume of the crystal), δ may be written:

$$\delta = 0.0794 \frac{\lambda^3}{V^2} \frac{1}{\sin 2\theta} \frac{\cos^2 2\theta_M + \cos^4 2\theta}{\cos^2 2\theta_M + \cos^2 2\theta}$$

Now, rewrite the relationship for the structure factor and considering the scale factor K, we can write:

$$|F_c^*| = K|F_c|\left(1 + 2r^*|F_c|^2\delta\right)^{-1/4} = K|F_c|\left(1 + 2r^*|F_c|^2 0.0794 \frac{\lambda^3}{V^2} \frac{1}{\sin 2\theta} \frac{pol1 + \cos^4 2\theta}{pol1 + \cos^2 2\theta}\right)^{-1/4}$$

In SHELX programs this expression is approximated by:

$$|F_c^*| = K|F_c|\left(1 + 0.001r^*|F_c|^2 \lambda^3 \frac{1}{\sin 2\theta}\right)^{-1/4}$$

Now, we calculate the derivative of $|F_c^*|$ respect to $|F_c|$:

$$\frac{\partial|F_c^*|}{\partial|F_c|} = K \frac{\left(1 + 2r^*|F_c|^2\delta\right) - |F_c|\frac{1}{4}\left(4r^*|F_c|\delta\right)}{\left(1 + 2r^*|F_c|^2\delta\right)^{5/4}} = K \frac{\left(1 + r^*|F_c|^2\delta\right)}{\left(1 + 2r^*|F_c|^2\delta\right)^{5/4}}$$

The partial derivatives become:

Atomic parameters:

$$\frac{\partial|F_c^*|}{\partial p_i} = \frac{\partial|F_c^*|}{\partial|F_c|}\frac{\partial|F_c|}{\partial p_i} = K \frac{\left(1 + r^*|F_c|^2\delta\right)}{\left(1 + 2r^*|F_c|^2\delta\right)^{5/4}} \frac{\partial|F_c|}{\partial p_i}$$

Overall scale factor K:

$$\frac{\partial|F_c^*|}{\partial K} = |F_c|\left(1 + 2r^*|F_c|^2\delta\right)^{-1/4}$$

Extiction parameter:

$$\frac{\partial|F_c^*|}{\partial r^*} = -\frac{1}{2}K|F_c^3|\delta\left(1 + 2r^*|F_c|^2\delta\right)^{-5/4}$$

VI. Refinement of Twinned Structures

Here we consider only twins by hemihedry (a particular case of merohedry) defined as regular aggregates consisting of only two microscopic domains of the same species whose lattices differ in orientation. In this case, we need an orientation matrix, **R** (the twinning law) and the fractional contribution, x ,of the smaller domain. Yeates (1997) described a method to estimate the twinning fraction. For more information on twinned structures please see Giacovazzo, (1992) and also visit the web site:
http://www.lcm3b.uhp-nancy.fr/mathcryst/twins.htm
For the refinement of twinned structures, see also Herbest-Irmer & Sheldrick, (1998).
We follow the procedure as described by H D Flack,(1983), considering the enantiomorph-polarity parameter x a special case of twinned crystal.

$$\left|F_{h,x}\right|^2 = (1-x)\left|F_h\right|^2 + x\left|F_{Rh}\right|^2$$

Rewrite the relationships of the structure factors:

$$F_h = A_h + iB_h = \left|F_h\right|\exp(i\phi)$$

$$\left|F_h\right| = A_h\cos\phi + B_h\sin\phi$$

and

$$F_{Rh} = A_{Rh} + iB_{Rh} = \left|F_{Rh}\right|\exp(i\phi)$$

$$\left|F_{Rh}\right| = A_{Rh}\cos\phi + B_{Rh}\sin\phi$$

The function being minimized by the least-squares method becomes:

$$M = \sum_h w_h\left(\left|F_h\right|^2_{oss} - K^2\left|F_{h,x}\right|^2\right)^2 = \sum_h w_h\left(\left|F_h\right|^2_{oss} - K^2\left[(1-x)\left|F_h\right|^2 + x\left|F_{Rh}\right|^2\right]\right)^2$$

$$M = \sum_h w_h\left(\left|F_h\right|^2_{oss} - K^2\left[(1-x)\left|F_h\right|^2 + x\left|F_{Rh}\right|^2\right] - K^2\sum_j\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_j}\right]\Delta r_j\right)^2 =$$

$$= \sum_h w_h\left(\Delta F^2{}_{h,x} - K^2\sum_j\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_j}\right]\Delta r_j\right)^2$$

Setting to zero the derivates of M with respect to the Δr_i .

$$\frac{\partial M}{\partial \Delta r_i} = \sum_h w_h\left(\Delta F^2{}_{h,x} - K^2\sum_j\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_j}\right]\Delta r_j\right)$$

$$K^2\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_i} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_i}\right] = 0$$

the normal equations are obtained:

$$\sum_h w_h K^2\left(\sum_j\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_j}\right]\Delta r_j\right)K^2\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_i} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_i}\right] =$$

$$= \sum_h w_h\Delta F^2{}_{h,x}K^2\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_i} + x2\left|F_{Rh}\right|\frac{\partial\left|F_{Rh}\right|}{\partial r_i}\right]$$

Considering:

$$\left|F_{h,x}\right|^2 = (1-x)\left(A_h\cos\phi_h + B_h\sin\phi_h\right)^2 + x\left(A_{Rh}\cos\phi_{Rh} + B_{Rh}\sin\phi_{Rh}\right)^2$$

we have:

$$\frac{\partial |F_{h,x}|^2}{\partial p_j} = \left((1-x)2|F_h| \frac{\partial |F_h|}{\partial p_j} + x2|F_{Rh}| \frac{\partial |F_{Rh}|}{\partial p_j} \right)$$

$$\frac{\partial |F_h|}{\partial p_j} = \frac{\partial A_h}{\partial p_j} \cos\phi_h + \frac{\partial B_h}{\partial p_j} \sin\phi_h$$

$$\frac{\partial |F_{Rh}|}{\partial p_j} = \frac{\partial A_{Rh}}{\partial p_j} \cos\phi_{Rh} + \frac{\partial B_{Rh}}{\partial p_j} \sin\phi_{Rh}$$

Atomic coordinates $\mathbf{r_j}$:

$$\frac{\partial A_{h,j}}{\partial \mathbf{r_j}} = -(2\pi \mathbf{h}) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[-\mathrm{B_j} \left(\frac{\sin\vartheta_h}{\lambda} \right)^2 \right] (occ_j) = -2\pi \mathbf{h} B_{h,j}$$

$$\frac{\partial A_{Rh,j}}{\partial \mathbf{r_j}} = -(2\pi \mathbf{Rh}) f_j \sin(2\pi \cdot \mathbf{Rh} \cdot \mathbf{r_j}) \exp\left[-\mathrm{B_j} \left(\frac{\sin\vartheta_h}{\lambda} \right)^2 \right] (occ_j) = -2\pi \mathbf{Rh} B_{Rh,j}$$

$$\frac{\partial B_{h,j}}{\partial \mathbf{r_j}} = (2\pi \mathbf{h}) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp\left[-\mathrm{B_j} \left(\frac{\sin\vartheta_h}{\lambda} \right)^2 \right] (occ_j) = 2\pi \mathbf{h} A_{h,j}$$

$$\frac{\partial B_{Rh,j}}{\partial \mathbf{r_j}} = (2\pi \mathbf{Rh}) f_j \cos(2\pi \cdot \mathbf{Rh} \cdot \mathbf{r_j}) \exp\left[-\mathrm{B_j} \left(\frac{\sin\vartheta_h}{\lambda} \right)^2 \right] (occ_j) = 2\pi \mathbf{Rh} A_{Rh,j}$$

Atomic isotropic thermal parameters B_j :

$$\frac{\partial A_{h,j}}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda} \right)^2 A_{h,j}$$

$$\frac{\partial A_{Rh,j}}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda} \right)^2 A_{Rh,j}$$

$$\frac{\partial B_{h,j}}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda} \right)^2 B_{h,j}$$

$$\frac{\partial B_{Rh,j}}{\partial \mathrm{B_j}} = -\left(\frac{\sin\vartheta_h}{\lambda} \right)^2 B_{Rh,j}$$

Atomic anisotropic thermal parameters β_{ij} :

$$\frac{\partial A_{h,j}}{\partial \beta_{(ij)_j}} = (-h_i h_j) f_j \cos(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}] (occ_j) = (-h_i h_j) A_{h,j}$$

$$\frac{\partial A_{Rh,j}}{\partial \beta_{(ij)_j}} = (-Rh_i Rh_j) f_j \cos(2\pi \cdot \mathbf{Rh} \cdot \mathbf{r_j}) \exp[-\mathbf{Rh}\beta_{ij}\mathbf{Rh}] (occ_j) = (-Rh_i Rh_j) A_{Rh,j}$$

$$\frac{\partial B_{h,j}}{\partial \beta_{(ij)_j}} = (-h_i h_j) f_j \sin(2\pi \cdot \mathbf{h} \cdot \mathbf{r_j}) \exp[-\mathbf{h}\beta_{ij}\mathbf{h}] (occ_j) = (-h_i h_j) B_{h,j}$$

$$\frac{\partial B_{Rh,j}}{\partial \beta_{(ij)_j}} = (-Rh_i Rh_j) f_j \sin(2\pi \cdot \mathbf{Rh} \cdot \mathbf{r_j}) \exp[-\mathbf{Rh}\beta_{ij}\mathbf{Rh}] (occ_j) = (-Rh_i Rh_j) B_{Rh,j}$$

Site occupation factors occ_j :

$$\frac{\partial A_{h,j}}{\partial occ_j} = \frac{A_{h,j}}{occ_j}$$

$$\frac{\partial A_{Rh,j}}{\partial occ_j} = \frac{A_{Rh,j}}{occ_j}$$

$$\frac{\partial B_{h,j}}{\partial occ_j} = \frac{B_{h,j}}{occ_j}$$

$$\frac{\partial B_{Rh,j}}{\partial occ_j} = \frac{B_{Rh,j}}{occ_j}$$

Twin fractional parameter x:

$$\frac{\partial \left|F_{h,x}\right|^2}{\partial x} = -\left|F_h\right|^2 + \left|F_{Rh}\right|^2$$

all the previous derivatives have to be multiplied by $(w_h K^2)$.

VII. Enantiomorph-polarity parameter

The enantiomorph-polarity parameter x for the observed intensity of the reflection h (as defined by H D Flack, 1983) is:

$$\left|F_{h,x}\right|^2 = (1-x)\left|F_h\right|^2 + x\left|F_{-h}\right|^2$$

where $|F_h|^2$ and $|F_{-h}|^2$ are the intensity of the two enantiomers.

The function being minimized by the least-squares method becomes:

$$M = \sum_h w_h \left(\left|F_h\right|^2_{oss} - K^2\left|F_{h,x}\right|^2\right)^2 = \sum_h w_h \left(\left|F_h\right|^2_{oss} - K^2\left[(1-x)\left|F_h\right|^2 + x\left|F_{-h}\right|^2\right]\right)^2$$

$$M = \sum_h w_h \left(\left|F_h\right|^2_{oss} - K^2\left[(1-x)\left|F_h\right|^2 + x\left|F_{-h}\right|^2\right] - K^2\sum_j \left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{-h}\right|\frac{\partial\left|F_{-h}\right|}{\partial r_j}\right]\Delta r_j\right)^2 =$$

$$= \sum_h w_h \left(\Delta F^2{}_{h,x} - K^2\sum_j \left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{-h}\right|\frac{\partial\left|F_{-h}\right|}{\partial r_j}\right]\Delta r_j\right)^2$$

Setting to zero the derivates of M with respect to the Δr_i .

$$\frac{\partial M}{\partial \Delta r_i} = \sum_h w_h \left(\Delta F^2{}_{h,x} - K^2\sum_j \left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_j} + x2\left|F_{-h}\right|\frac{\partial\left|F_{-h}\right|}{\partial r_j}\right]\Delta r_j\right)$$

$$K^2\left[(1-x)2\left|F_h\right|\frac{\partial\left|F_h\right|}{\partial r_i} + x2\left|F_{-h}\right|\frac{\partial\left|F_{-h}\right|}{\partial r_i}\right] = 0$$

the normal equations are obtained:

$$\sum_h w_h K^2 \left(\sum_j \left[(1-x)2|F_h| \frac{\partial |F_h|}{\partial r_j} + x2|F_{-h}| \frac{\partial |F_{-h}|}{\partial r_j} \right] \Delta r_j \right) K^2 \left[(1-x)2|F_h| \frac{\partial |F_h|}{\partial r_i} + x2|F_{-h}| \frac{\partial |F_{-h}|}{\partial r_i} \right] =$$

$$= \sum_h w_h \Delta F^2{}_{h,x} K^2 \left[(1-x)2|F_h| \frac{\partial |F_h|}{\partial r_i} + x2|F_{-h}| \frac{\partial |F_{-h}|}{\partial r_i} \right]$$

We consider now the anomalous scattering effect, and as we have seen before we have:

for *h*:

$$F_h = (A_h - D_h) + i(B_h + C_h)$$

$$\tan \phi_h = \frac{(B_h + C_h)}{(A_h - D_h)}$$

$$|F_h| = (A_h - D_h)\cos \phi_h + (B_h + C_h)\sin \phi_h$$

and for –*h*:

$$F_{-h} = (A_h + D_h) - i(B_h - C_h)$$

$$\tan \phi_{-h} = \frac{(-B_h + C_h)}{(A_h + D_h)}$$

$$|F_{-h}| = (A_h + D_h)\cos \phi_{-h} + (-B_h + C_h)\sin \phi_{-h}$$

The structure factor now is:

$$|F_{h,x}|^2 = (1-x)[(A_h - D_h)\cos \phi_h + (B_h + C_h)\sin \phi_h]^2 + x[(A_h + D_h)\cos \phi_{-h} + (-B_h + C_h)\sin \phi_{-h}]^2$$

we have:

$$\frac{\partial |F_{h,x}|^2}{\partial p_j} = \left((1-x)2|F_h| \frac{\partial |F_h|}{\partial p_j} + x2|F_{-h}| \frac{\partial |F_{-h}|}{\partial p_j} \right)$$

$$\frac{\partial |F_h|}{\partial p_j} = \left(\frac{\partial A_h}{\partial p_j} - \frac{\partial D_h}{\partial p_j} \right) \cos \phi_h + \left(\frac{\partial B_h}{\partial p_j} + \frac{\partial C_h}{\partial p_j} \right) \sin \phi_h$$

$$\frac{\partial |F_{-h}|}{\partial p_j} = \left(\frac{\partial A_h}{\partial p_j} + \frac{\partial D_h}{\partial p_j} \right) \cos \phi_{-h} + \left(-\frac{\partial B_h}{\partial p_j} + \frac{\partial C_h}{\partial p_j} \right) \sin \phi_{-h}$$

Flack parameter *x*:

$$\frac{\partial |F_{h,x}|^2}{\partial x} = -|F_h|^2 + |F_{-h}|^2$$

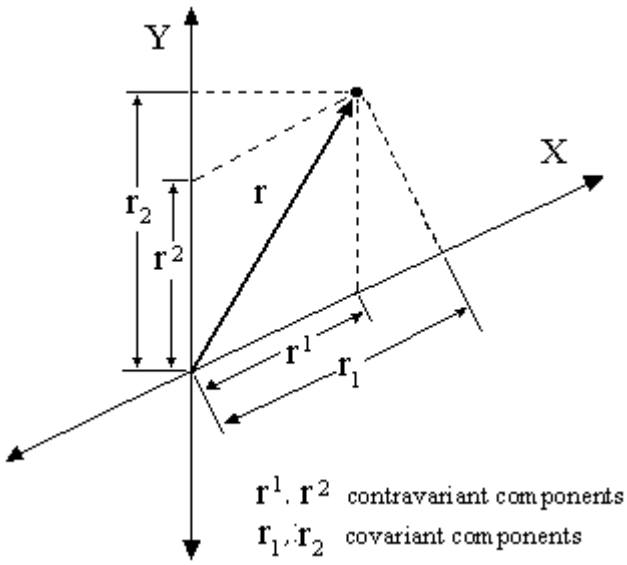
VIII. Metric and vector notation

We assumed the tensor notation according to the Levi-Civita notation. A crystallographic cell in the real space (**a,b,c,α,β,γ)** with volume V is written as $\mathbf{a}_i$, while the reciprocal cell is $\mathbf{a}^i$ and the metric tensors $\mathbf{G}_{ij}$ and $\mathbf{G}^{ij}$, direct and reciprocal metric tensors respectively, are defined as following:

$$\mathbf{G}_{ij} = \begin{pmatrix} a^2 & ab\cos\gamma & ac\cos\beta \\ ab\cos\gamma & b^2 & bc\cos\alpha \\ ac\cos\beta & bc\cos\alpha & c^2 \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix}$$

$$\mathbf{G}^{ij} = \begin{pmatrix} a^{*2} & a^*b^*\cos\gamma^* & a^*c^*\cos\beta^* \\ a^*b^*\cos\gamma^* & b^{*2} & b^*c^*\cos\alpha^* \\ a^*c^*\cos\beta^* & b^*c^*\cos\alpha^* & c^{*2} \end{pmatrix} = \begin{pmatrix} g^{11} & g^{12} & g^{13} \\ g^{21} & g^{22} & g^{23} \\ g_{31} & g^{32} & g^{33} \end{pmatrix}$$

With respect to the $\mathbf{a}_i$ base, vectors are written as $\mathbf{r}^i = \left(r^x, r^y, r^z\right)$, and with respect to the $\mathbf{a}^i$ base, vectors are written as $\mathbf{r}_i = \left(r_x, r_y, r_z\right)$, *i.e.* a vector **r** can be represented by the $\mathbf{r}^i$ or $\mathbf{r}_i$, contravariant and covariant components respectively:



The squared length of **r** can be expressed in terms of his components taking into account the metric tensor with Einstein's summation convention as follows:

$$|r|^2 = \mathbf{G}_{ij}\mathbf{r}^i\mathbf{r}^j = \mathbf{G}^{ij}\mathbf{r}_i\mathbf{r}_j$$

and it easy to show that $\mathbf{r}_i = \mathbf{G}_{ij}\mathbf{r}^j$ and if we perform the inverse operation, we have $\mathbf{r}^i = \mathbf{G}^{ij}\mathbf{r}_j$ with $\mathbf{G}_{ij} = \left(\mathbf{G}^{ij}\right)^{-1}$ and then we obtain

$$|r|^2 = \mathbf{r}_i\mathbf{r}^j$$

If the coordinate system is a orthogonal monometric base ($\mathbf{G}_{ij} = \mathbf{G}^{ij} = \mathbf{I}$), the contravariant and covariant components are identical:

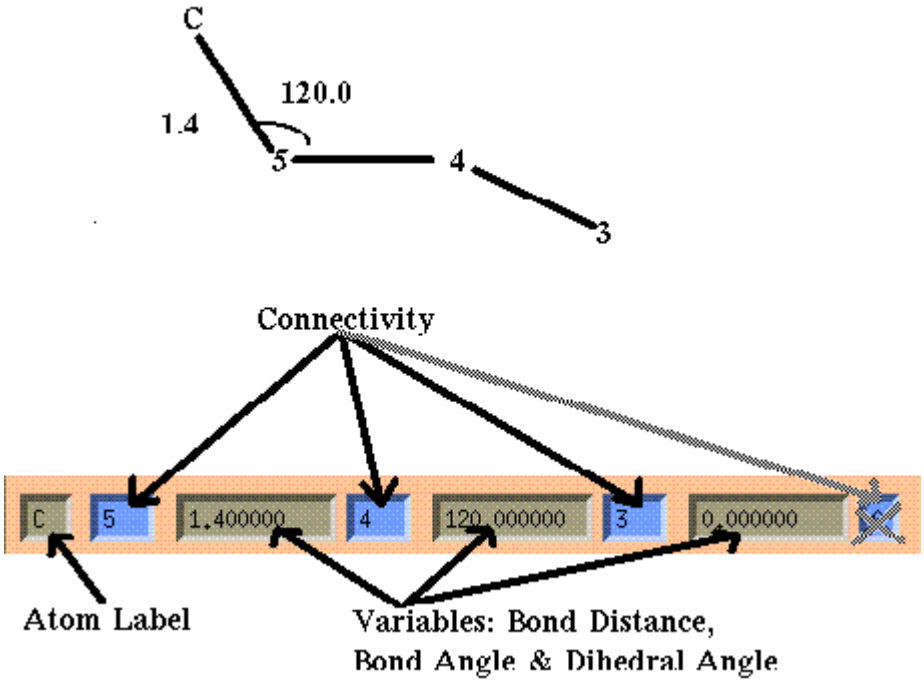
$$\mathbf{r}^j = \mathbf{r}_i$$

IX. Rigid body constraint

A rigid body is formed by a number of atoms having known fixed geometry. Our approach follows the traditional refinement of the centroid and of three Eulerian angles. In fact, the position and the orientation of a rigid body may be described by three positional and three rotational parameters. Therefore, for each rigid body of *n* atoms the number of independent positional parameters reduces from 3*n* to 6.

The calculations could be performed according to the following scheme.

1) Define a rigid body in a Z-matrix representation. In the Z-matrix approach used to define connectivity between atoms in a molecule the position of each atom (except the first three) is defined with respect to three previously defined atoms. The parameters one needs are distances (*d*), angles (α) and dihedral angles (γ).
   In this example



- a **Carbon** atom
- is connected to atom number **5** with a *Bond Distance* of **1.4** Angstroms,
- The vector from the **Carbon** atom to atom number **5** and the vector from atom number **5** to atom number **4** make a *Bond Angle* of **120.0** degrees.
- The two planes constituted by atoms (Carbon-5-4) and (5-4-3) make an *Dihedral/torsion angle* of **0.0** degrees.

The *first* atom in the Z-matrix has no previously defined atoms to refer to and is the origin, the *second* atom is always connected to the first atom by just a bond distance, and the *third* atom is connected to the first two by just a bond distance and a bond angle. The vector (**2-1**) defines the **x** axis , the plane which contains (**1-2-3**) define where the **y** axis lies, and then the **z** axis is obtained as vector perpendicular to that plane.
A fine Z-matrix editor can be found at
http://www.cmbi.ru.nl/molden/zmat/zmat.html

For a benzene group, the Z-matrix representation is:

Atom Name	Atom Connect	Bond Distance	Angle Connect	Bond Angle	Dihedral Connect	Dihedral Angle
C1						
C2	1	1.38				
C3	2	1.38	1	120.0		
C4	3	1.38	2	120.0	1	0.00
C5	4	1.38	3	120.0	2	0.00
C6	5	1.38	4	120.0	3	0.00

2) Transform the coordinates in Z-matrix representation in its local Cartesian frame **x** (in Ångstroms). In this case, the base $\mathbf{a}_i$ is orthogonal and the metric tensor $\mathbf{G}^{ij} = \mathbf{I}$, therefore we do not considered the metric tensor in the calculation. The general algorithm is the following:

$\mathbf{r}_1 = (0,0,0)$

$\mathbf{r}_2 = (d_2,0,0)$

For the atom $\mathbf{r}_3$:

Calculate the versor $\mathbf{p}^i$ along ***x***-axis as follows:

$\mathbf{u}^i = \mathbf{r}_2 - \mathbf{r}_1$

$|u|^2 = \mathbf{u}^i \bullet \mathbf{u}^i$

$u = \sqrt{|u|^2}$

$\mathbf{p}^i = \dfrac{\mathbf{u}^i}{u}$

Calculate the versor $\mathbf{q}^i$ along ***z***-axis:

$\mathbf{v}^i = \mathbf{p}^i \wedge (0,1,0)$

$\mathbf{q}^i = \dfrac{\mathbf{v}^i}{v}$

Calculate the versor $\mathbf{s}^i$ along ***y***-axis:

$\mathbf{s}^i = \mathbf{q}^i \wedge \mathbf{p}^i$

then we can calculate the position for the atom $\mathbf{r}_3$

$\mathbf{r}_3 = \mathbf{r}_2 - \mathbf{p}^i d_3 \cos\alpha_3 + \mathbf{s}^i d_3 \sin\alpha_3$

For the next atom i (atom connect j, bond distance d_i, angle connect k, bond angle α_i, dihedral connect l, dihedral angle ψ_i):

Calculate the versor $\mathbf{p}^i$ along k-j direction to establish the local x-axis:

$\mathbf{u}^i = \mathbf{r}_j - \mathbf{r}_k$

$\mathbf{p}^i = \dfrac{\mathbf{u}^i}{u}$

Calculate the vector along l-k direction:

$\mathbf{w}^i = \mathbf{r}_l - \mathbf{r}_k$

Calculate the cross product of $\mathbf{w}^i$ and $\mathbf{p}^i$ to establish the versor $\mathbf{q}^i$ along the local z-axis:

$\mathbf{v}^i = \mathbf{p}^i \wedge \mathbf{w}^i$

$\mathbf{q}^i = \dfrac{\mathbf{v}^i}{v}$

Calculate the cross product of $\mathbf{p}^i$ and $\mathbf{q}^i$ to establish the versor $\mathbf{s}^i$ along the local *y*-axis:

$$\mathbf{s}^i = \mathbf{q}^i \wedge \mathbf{p}^i$$

$$\mathbf{r}_i = \mathbf{r}_j - \mathbf{p}^i d_i \cos\alpha_i + \mathbf{s}^i d_i \sin\alpha_i \cos\psi_i + \mathbf{q}^i d_i \sin\alpha_i \sin\psi_i$$

3) If the matrix $\mathbf{G}=\mathbf{U}^{-1}\mathbf{R}$ connects the local Cartesian system to the crystallographic system, where **U** is the orthogonalization matrix of the crystallographic cell and **R** is orientation matrix (the product of three rotation matrix), we have:

$$\mathbf{x} = \mathbf{x}^0 + \mathbf{GX}$$

where **x** are the coordinates of atoms and $\mathbf{x}^0$ are the coordinates of the origin of the group in the crystallographic cell, **X** are the coordinates of atoms in the local Cartesian system of the group.

4) The contribution F_h^G to the structure factors is given by:

$$F_h^G = \sum_{j=1}^{n} f_j \exp\left(2\pi i \cdot \mathbf{h} \cdot \mathbf{x_j}\right) \exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right](occ_j)$$

$$= \sum_{j=1}^{n} f_j \exp\left(2\pi i \cdot \mathbf{h} \cdot \left(\mathbf{x}^0 + \mathbf{GX}\right)_\mathbf{j}\right) \exp\left[-\mathrm{B_j}\left(\frac{\sin\vartheta_h}{\lambda}\right)^2\right](occ_j)$$

5) Define the derivatives with respect to the rigid body parameters ($x_0,y_0,z_0,\theta,\psi,\varphi$):

$$\frac{\partial\left|F_h^G\right|}{\partial p_j} = \frac{\partial A_h^G}{\partial p_j}\cos\phi + \frac{\partial B_h^G}{\partial p_j}\sin\phi$$

Atomic coordinates shift $\mathbf{x_0}$:

$$\frac{\partial\left|A_h^G\right|}{\partial \mathbf{x_o}} = \sum_{j=1}^{n}\frac{\partial A_j}{\partial \mathbf{x}_j}$$

$$\frac{\partial\left|B_h^G\right|}{\partial \mathbf{x_o}} = \sum_{j=1}^{n}\frac{\partial B_j}{\partial \mathbf{x}_j}$$

Rotational parameters θ,ψ,φ:

$$\frac{\partial\left|A_h^G\right|}{\partial\theta} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\theta} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{GX}_{ij}}{\partial\theta} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\theta}\mathbf{X}_{ij}$$

$$\frac{\partial\left|B_h^G\right|}{\partial\theta} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\theta} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\theta}\mathbf{X}_{ij}$$

$$\frac{\partial\left|A_h^G\right|}{\partial\psi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\psi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\psi}\mathbf{X}_{ij}$$

$$\frac{\partial\left|B_h^G\right|}{\partial\psi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\psi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\psi}\mathbf{X}_{ij}$$

$$\frac{\partial\left|A_h^G\right|}{\partial\varphi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\varphi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial A_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\varphi}\mathbf{X}_{ij}$$

$$\frac{\partial\left|B_h^G\right|}{\partial\varphi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\frac{\partial \mathbf{x}_{ji}}{\partial\varphi} = \sum_{j=1}^{n}\sum_{i=1}^{3}\frac{\partial B_j}{\partial \mathbf{x}_{ji}}\mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial\varphi}\mathbf{X}_{ij}$$

6) To calculate the derivative respect the rotational parameters, we have to define the orientation matrix (Andreev et al.,1997):

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\varphi - \cos\theta\sin\varphi\sin\psi & \cos\psi\sin\varphi + \cos\theta\cos\varphi\sin\psi & \sin\theta\sin\psi \\ -\sin\psi\cos\varphi - \cos\theta\sin\varphi\cos\psi & -\sin\psi\sin\varphi + \cos\theta\cos\varphi\cos\psi & \cos\psi\sin\theta \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\theta \end{bmatrix}$$

and the orthogonalization matrix from the cell parameters $a,b,c,\alpha,\beta,\gamma$:

$$\mathbf{U} = \begin{bmatrix} a\sin\beta & -b\sin\alpha\cos\gamma^* & 0 \\ 0 & b\sin\alpha\sin\gamma^* & 0 \\ a\cos\beta & b\cos\alpha & c \end{bmatrix}$$

Then, we calculate an approximate **G** matrix from the relationship

$\mathbf{x} = \mathbf{x}^0 + \mathbf{GX}$

where **x**, $\mathbf{x^0}$ and **X** are known and we obtain the R matrix from:

$\mathbf{R} = \mathbf{UG}$

and values of the Eulerian angles can be readily calculated as follows:

$\theta = \text{acos}(R_{33})$

$$\begin{cases} \sin\varphi = \left[\dfrac{R_{31}}{\sqrt{(1-R_{33}{}^2)}}\right] \\ \cos\varphi = -\left[\dfrac{R_{32}}{\sqrt{(1-R_{33}{}^2)}}\right] \end{cases}$$

$$\begin{cases} \sin\psi = \left[\dfrac{R_{13}}{\sqrt{(1-R_{33}{}^2)}}\right] \\ \cos\psi = \left[\dfrac{R_{23}}{\sqrt{(1-R_{33}{}^2)}}\right] \end{cases}$$

giving the initial value of θ,φ and ψ .

Now we can update the **x** coordinates of the atoms of the group, respect to the coordinates **X** of the model, knowing the Eulerian angles:

$\mathbf{G} = \mathbf{U}^{-1}\mathbf{R}$

$\mathbf{x} = \mathbf{x}^0 + \mathbf{GX}$

Calculate the derivatives of the **R** matrix for the Eulerian angles:

$$\frac{\partial\mathbf{R}}{\partial\theta} = \begin{bmatrix} \sin\theta\sin\varphi\sin\psi & -\sin\theta\cos\varphi\sin\psi & \cos\theta\sin\psi \\ \sin\theta\sin\varphi\cos\psi & -\sin\theta\cos\varphi\cos\psi & \cos\psi\cos\theta \\ \cos\theta\sin\varphi & -\cos\theta\cos\varphi & -\sin\theta \end{bmatrix}$$

$$\frac{\partial \mathbf{R}}{\partial \varphi} = \begin{bmatrix} -\cos\psi\sin\varphi - \cos\theta\cos\varphi\sin\psi & \cos\psi\cos\varphi - \cos\theta\sin\varphi\sin\psi & 0 \\ \sin\psi\sin\varphi - \cos\theta\cos\varphi\cos\psi & -\sin\psi\cos\varphi - \cos\theta\sin\varphi\cos\psi & 0 \\ \sin\theta\cos\varphi & \sin\theta\sin\varphi & 0 \end{bmatrix}$$

$$\frac{\partial \mathbf{R}}{\partial \psi} = \begin{bmatrix} -\sin\psi\cos\varphi - \cos\theta\sin\varphi\cos\psi & -\sin\psi\sin\varphi + \cos\theta\cos\varphi\cos\psi & \sin\theta\cos\psi \\ -\cos\psi\cos\varphi + \cos\theta\sin\varphi\sin\psi & -\cos\psi\sin\varphi - \cos\theta\cos\varphi\sin\psi & -\sin\psi\sin\theta \\ 0 & 0 & 0 \end{bmatrix}$$

obtaining:

$$\frac{\partial \mathbf{x}}{\partial \theta} = \mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial \theta}\mathbf{X}$$

$$\frac{\partial \mathbf{x}}{\partial \varphi} = \mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial \varphi}\mathbf{X}$$

$$\frac{\partial \mathbf{x}}{\partial \psi} = \mathbf{U}^{-1}\frac{\partial \mathbf{R}}{\partial \psi}\mathbf{X}$$

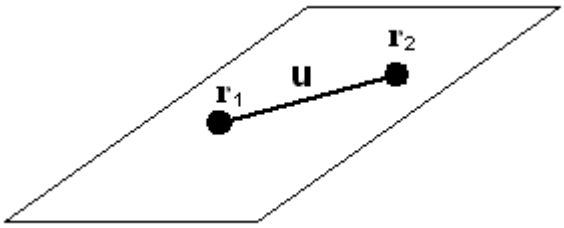
A good collection of useful routines for this argument can be found at the DYNAMO site:

http://www.ibs.fr/ext/labos/LDM/projet6/Main.html

DYNAMO is a library of Fortran 90 modules that has been designed for the simulation of molecular systems using molecular mechanical (MM) and hybrid quantum mechanical (QM)/MM potential energy functions.

X. Bond distance restraint

The bond distance is the calculated distance between the atom A and the atom B . The positions of the atoms (A, B) are represented by $\mathbf{r_n}$ =(x_n,y_n,z_n) (n = 1,2)



and the inter-atomic vectors is:

$$\mathbf{u}^i = \mathbf{r_1} - \mathbf{r_2} \rightarrow \left(u^1, u^2, u^3\right) = \left[(x_1 - x_2), (y_1 - y_2), (z_1 - z_2)\right]$$

The interatomic distance is obtained by:

$$|u|^2 = \mathbf{u}^i \bullet \mathbf{G}_{ij}\mathbf{u}^j = \begin{Bmatrix} g_{11}(x_1 - x_2)^2 & + g_{12}(x_1 - x_2)(y_1 - y_2) & + g_{13}(x_1 - x_2)(z_1 - z_2) \\ + g_{21}(y_1 - y_2)(x_1 - x_2) & + g_{22}(y_1 - y_2)^2 & + g_{23}(y_1 - y_2)(z_1 - z_2) \\ + g_{31}(z_1 - z_2)(x_1 - x_2) & + g_{32}(z_1 - z_2)(y_1 - y_2) & + g_{33}(z_1 - z_2)^2 \end{Bmatrix}$$

$$u = \sqrt{|u|^2}$$

The derivatives of *u* with respect to *r* is :

$$\frac{\partial u}{\partial r} = \frac{1}{2u} \cdot \partial(\mathbf{u}^i \bullet \mathbf{G}_{ij}\mathbf{u}^j) = \frac{1}{2u} \cdot \left(2G_{ij}u^j\right) = \frac{1}{u}G_{ij}u^j$$

Then we can calculate the derivatives of *u* with respect to the coordinates *x,y,z* of the two atoms.

Atom 1

Parameter x_1

$$\frac{\partial u}{\partial x_1} = \frac{1}{u} \cdot \left[g_{11}(x_1 - x_2) + g_{12}(y_1 - y_2) + g_{13}(z_1 - z_2)\right]$$

Parameter y_1

$$\frac{\partial u}{\partial y_1} = \frac{1}{u} \cdot \left[g_{21}(x_1 - x_2) + g_{22}(y_1 - y_2) + g_{23}(z_1 - z_2)\right]$$

Parameter z_1

$$\frac{\partial u}{\partial z_1} = \frac{1}{u} \cdot \left[g_{31}(x_1 - x_2) + g_{32}(y_1 - y_2) + g_{33}(z_1 - z_2)\right]$$

Atom 2

Parameter x_2

$$\frac{\partial u}{\partial x_2} = -\frac{1}{u} \cdot \left[g_{11}(x_1 - x_2) + g_{12}(y_1 - y_2) + g_{13}(z_1 - z_2)\right] = -\frac{\partial u}{\partial x_1}$$

Parameter y_2

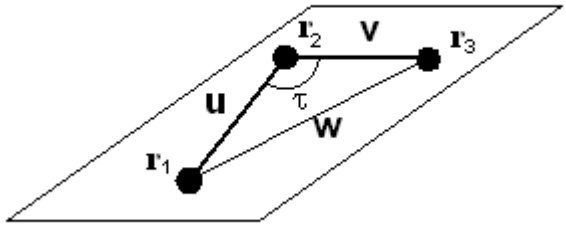
$$\frac{\partial u}{\partial y_2} = -\frac{1}{u}\cdot\left[g_{21}(x_1 - x_2) + g_{22}(y_1 - y_2) + g_{23}(z_1 - z_2)\right] = -\frac{\partial u}{\partial y_1}$$

Parameter z_2

$$\frac{\partial u}{\partial z_2} = -\frac{1}{u}\cdot\left[g_{31}(x_1 - x_2) + g_{32}(y_1 - y_2) + g_{33}(z_1 - z_2)\right] = -\frac{\partial u}{\partial z_1}$$

XI. Bond angle restraint

The bond angle $\boldsymbol{\tau}$ at the atom B is the angle between the bond A—B and the bond B—C. The positions of the atoms (A, B,C) are represented by $\mathbf{r_n} = (x_n, y_n, z_n)$ (n = 1,2,3)



and the inter-atomic vectors are:

$$\mathbf{u}^i = \mathbf{r_1} - \mathbf{r_2} \rightarrow \left(u^1, u^2, u^3\right) = \left[(x_1 - x_2), (y_1 - y_2), (z_1 - z_2)\right]$$
$$\mathbf{v}^i = \mathbf{r_3} - \mathbf{r_2} \rightarrow \left(v^1, v^2, v^3\right) = \left[(x_3 - x_2), (y_3 - y_2), (z_3 - z_2)\right]$$
$$(\mathbf{u} - \mathbf{v})^i = \mathbf{w}^i = \mathbf{r_1} - \mathbf{r_3} \rightarrow \left(w^1, w^2, w^3\right) = \left[(x_1 - x_3), (y_1 - y_3), (z_1 - z_3)\right]$$

We have the relationships :

$$|\mathbf{u} - \mathbf{v}|^2 = |\mathbf{w}|^2 = |\mathbf{u}|^2 + |\mathbf{v}|^2 - 2|\mathbf{u}||\mathbf{v}|\cos\boldsymbol{\tau}$$
$$\cos\boldsymbol{\tau} = \frac{|\mathbf{u}|^2 + |\mathbf{v}|^2 - |\mathbf{w}|^2}{2|\mathbf{u}||\mathbf{v}|}$$

and the angle $\boldsymbol{\tau}$ between the two vectors **u** and **v** is obtained:

$$\boldsymbol{\tau} = \operatorname{acos}\left(\frac{|\mathbf{u}|^2 + |\mathbf{v}|^2 - |\mathbf{w}|^2}{2|\mathbf{u}||\mathbf{v}|}\right)$$

The interatomic distances are:

$$|u|^2 = \mathbf{u}^i \bullet \mathbf{G}_{ij}\mathbf{u}^j = \begin{Bmatrix} g_{11}(x_1 - x_2)^2 & + g_{12}(x_1 - x_2)(y_1 - y_2) & + g_{13}(x_1 - x_2)(z_1 - z_2) \\ + g_{21}(y_1 - y_2)(x_1 - x_2) & + g_{22}(y_1 - y_2)^2 & + g_{23}(y_1 - y_2)(z_1 - z_2) \\ + g_{31}(z_1 - z_2)(x_1 - x_2) & + g_{32}(z_1 - z_2)(y_1 - y_2) & + g_{33}(z_1 - z_2)^2 \end{Bmatrix}$$

$$u = \sqrt{|u|^2}$$

$$|v|^2 = \mathbf{v}^i \bullet \mathbf{G}_{ij} \mathbf{v}^j = \begin{Bmatrix} g_{11}(x_3 - x_2)^2 & + g_{12}(x_3 - x_2)(y_3 - y_2) & + g_{13}(x_3 - x_2)(z_3 - z_2) \\ + g_{21}(y_3 - y_2)(x_3 - x_2) & + g_{22}(y_3 - y_2)^2 & + g_{23}(y_3 - y_2)(z_3 - z_2) \\ + g_{31}(z_3 - z_2)(x_3 - x_2) & + g_{32}(z_3 - z_2)(y_3 - y_2) & + g_{33}(z_3 - z_2)^2 \end{Bmatrix}$$

$$v = \sqrt{|v|^2}$$

$$|w|^2 = \mathbf{w}^i \bullet \mathbf{G}_{ij} \mathbf{w}^j = \begin{Bmatrix} g_{11}(x_1 - x_3)^2 & + g_{12}(x_1 - x_3)(y_1 - y_3) & + g_{13}(x_1 - x_3)(z_1 - z_3) \\ + g_{21}(y_1 - y_3)(x_1 - x_3) & + g_{22}(y_1 - y_3)^2 & + g_{23}(y_1 - y_3)(z_1 - z_3) \\ + g_{31}(z_1 - z_3)(x_1 - x_3) & + g_{32}(z_1 - z_3)(y_1 - y_3) & + g_{33}(z_1 - z_3)^2 \end{Bmatrix}$$

$$w = \sqrt{|w|^2}$$

It is useful to rewrite the relationship to obtain the angle in the form:

$$\boldsymbol{\tau} = \operatorname{acos}\left(\frac{u^2 + v^2 - w^2}{2uv}\right) = \operatorname{acos}\left(\frac{u}{2v} + \frac{v}{2u} - \frac{w^2}{2uv}\right) = \operatorname{acos}(T)$$

Then using the chain rule, the derivatives of the function with respect to an atomic position *r* and a generic distance involved *d* are:

$$\frac{\partial \boldsymbol{\tau}}{\partial r} = -\frac{1}{\sin\tau} \sum \frac{\partial T}{\partial d} \frac{\partial d}{\partial r}$$

Considering the distance *u*, the derivatives are:

$$\frac{\partial T}{\partial u} = \frac{1}{2v} - \frac{v}{2u^2} + \frac{w^2}{2u^2 v} = \frac{u^2 - v^2 + w^2}{2u^2 v}$$

$$\frac{\partial u}{\partial r} = \frac{1}{2u} \cdot \partial(\mathbf{u}^i \bullet \mathbf{G}_{ij} \mathbf{u}^j) = \frac{1}{u} G_{ij} u^j$$

We obtain:

$$\frac{\partial \boldsymbol{\tau}}{\partial r} = -\frac{1}{\sin\tau}\left(\frac{u^2 - v^2 + w^2}{2u^2 v}\right)\frac{1}{u} G_{ij} u^j$$

Considering the distance *v*:

$$\frac{\partial T}{\partial v} = -\frac{u}{2v^2} + \frac{1}{2u} + \frac{w^2}{2uv^2} = \frac{-u^2 + v^2 + w^2}{2uv^2}$$

$$\frac{\partial v}{\partial r} = \frac{1}{2v} \cdot \partial(\mathbf{v}^i \bullet \mathbf{G}_{ij} \mathbf{v}^j) = \frac{1}{v} G_{ij} v^j$$

We obtain:

$$\frac{\partial \boldsymbol{\tau}}{\partial r} = -\frac{1}{\sin\tau}\left(\frac{-u^2 + v^2 + w^2}{2uv^2}\right)\frac{1}{v} G_{ij} v^j$$

Considering the distance *w*:

$$\frac{\partial T}{\partial w} = -\frac{2w}{2uv}$$

$$\frac{\partial w}{\partial r} = \frac{1}{2w} \cdot \partial(\mathbf{w}^i \bullet \mathbf{G}_{ij} \mathbf{w}^j) = \frac{1}{w} G_{ij} w^j$$

We obtain:

$$\frac{\partial \boldsymbol{\tau}}{\partial r} = \frac{1}{\sin\tau}\left(\frac{2w}{2uv}\right)\frac{1}{w} G_{ij} w^j = \frac{1}{\sin\tau}\frac{1}{uv} G_{ij} w^j$$

Now , we are able to calculate the derivatives with respect to the coordinates *x,y,z* of the three atoms involved.

$$\frac{\partial \boldsymbol{\tau}}{\partial r} = \frac{1}{\sin\tau}\left\{\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)G_{ij}u^j + \frac{1}{v}\left(\frac{u^2-v^2-w^2}{2uv^2}\right)G_{ij}v^j + \frac{1}{uv}\cdot G_{ij}w^j\right\}$$

Atom 1

Parameter x_1

$$\frac{\partial \boldsymbol{\tau}}{\partial x_1} = \frac{1}{\sin\tau}\left\{\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{11}(x_1-x_2)+g_{12}(y_1-y_2)+g_{13}(z_1-z_2)\right]+\right.$$

$$\left.+\frac{1}{uv}\left[g_{11}(x_1-x_3)+g_{12}(y_1-y_3)+g_{13}(z_1-z_3)\right]\right\}$$

Parameter y_1

$$\frac{\partial \boldsymbol{\tau}}{\partial y_1} = \frac{1}{\sin\tau}\left\{\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{21}(x_1-x_2)+g_{22}(y_1-y_2)+g_{23}(z_1-z_2)\right]+\right.$$

$$\left.+\frac{1}{uv}\left[g_{21}(x_1-x_3)+g_{22}(y_1-y_3)+g_{23}(z_1-z_3)\right]\right\}$$

Parameter z_1

$$\frac{\partial \boldsymbol{\tau}}{\partial z_1} = \frac{1}{\sin\tau}\left\{\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{31}(x_1-x_2)+g_{32}(y_1-y_2)+g_{33}(z_1-z_2)\right]+\right.$$

$$\left.+\frac{1}{uv}\left[g_{31}(x_1-x_3)+g_{32}(y_1-y_3)+g_{33}(z_1-z_3)\right]\right\}$$

Atom 2

Parameter x_2

$$\frac{\partial \boldsymbol{\tau}}{\partial x_2} = \frac{1}{\sin\tau}\left\{-\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{11}(x_1-x_2)+g_{12}(y_1-y_2)+g_{13}(z_1-z_2)\right]+\right.$$

$$\left.-\frac{1}{v}\left(\frac{u^2-v^2-w^2}{2uv^2}\right)\left[g_{11}(x_3-x_2)+g_{12}(y_3-y_2)+g_{13}(z_3-z_2)\right]\right\}$$

Parameter y_2

$$\frac{\partial \boldsymbol{\tau}}{\partial y_2} = \frac{1}{\sin\tau}\left\{-\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{21}(x_1-x_2)+g_{22}(y_1-y_2)+g_{23}(z_1-z_2)\right]+\right.$$

$$\left.-\frac{1}{v}\left(\frac{u^2-v^2-w^2}{2uv^2}\right)\left[g_{21}(x_3-x_2)+g_{22}(y_3-y_2)+g_{23}(z_3-z_2)\right]\right\}$$

Parameter z_2

$$\frac{\partial \boldsymbol{\tau}}{\partial z_2} = \frac{1}{\sin\tau}\left\{-\frac{1}{u}\left(\frac{-u^2+v^2-w^2}{2u^2v}\right)\left[g_{31}(x_1-x_2)+g_{32}(y_1-y_2)+g_{33}(z_1-z_2)\right]+\right.$$

$$\left.-\frac{1}{v}\left(\frac{u^2-v^2-w^2}{2uv^2}\right)\left[g_{31}(x_3-x_2)+g_{32}(y_3-y_2)+g_{33}(z_3-z_2)\right]\right\}$$

Atom 3

Parameter x_3

$$\frac{\partial \boldsymbol{\tau}}{\partial x_3} = \frac{1}{\sin\tau}\left\{\frac{1}{v}\left(\frac{u^2 - v^2 - w^2}{2uv^2}\right)\left[g_{11}(x_3 - x_2) + g_{12}(y_3 - y_2) + g_{13}(z_3 - z_2)\right] + \right.$$

$$\left. -\frac{1}{uv}\left[g_{11}(x_1 - x_3) + g_{12}(y_1 - y_3) + g_{13}(z_1 - z_3)\right]\right\}$$

Parameter y_3

$$\frac{\partial \boldsymbol{\tau}}{\partial y_3} = \frac{1}{\sin\tau}\left\{\frac{1}{v}\left(\frac{u^2 - v^2 - w^2}{2uv^2}\right)\left[g_{21}(x_3 - x_2) + g_{22}(y_3 - y_2) + g_{23}(z_3 - z_2)\right] + \right.$$

$$\left. -\frac{1}{uv}\left[g_{21}(x_1 - x_3) + g_{22}(y_1 - y_3) + g_{23}(z_1 - z_3)\right]\right\}$$

Parameter z_3

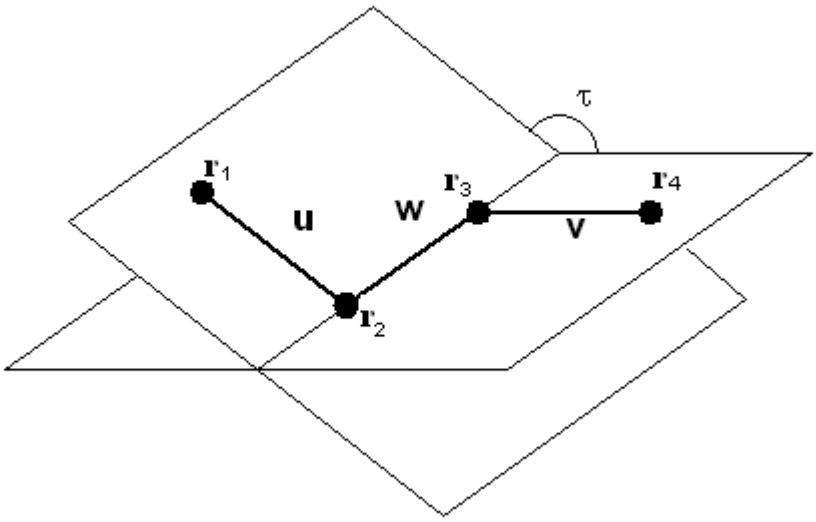
$$\frac{\partial \boldsymbol{\tau}}{\partial z_3} = \frac{1}{\sin\tau}\left\{\frac{1}{v}\left(\frac{u^2 - v^2 - w^2}{2uv^2}\right)\left[g_{31}(x_3 - x_2) + g_{32}(y_3 - y_2) + g_{33}(z_3 - z_2)\right] + \right.$$

$$\left. -\frac{1}{uv}\left[g_{31}(x_1 - x_3) + g_{32}(y_1 - y_3) + g_{33}(z_1 - z_3)\right]\right\}$$

XII. Torsion angle restraint

The torsion (or twist) angle **τ** about the line between atoms B--C of the sequence of four atoms A--B--C--D is defined as the rotation angle required to correspond the projection of the line B--A to the projection of the line C--D, seen along the direction of the line B--C. The sign of the angle is positive if the rotation is clockwise.

The positions of the atoms (A,B,C,D) are represented by $\mathbf{r_n}=(x_n,y_n,z_n)$ (n=1,2,3,4) and the interatomic vectors are:



$$\mathbf{u}^i = \mathbf{r_1} - \mathbf{r_2} \rightarrow \left(u^1, u^2, u^3\right) = \left[\left(x_1 - x_2\right), \left(y_1 - y_2\right), \left(z_1 - z_2\right)\right]$$
$$\mathbf{v}^i = \mathbf{r_4} - \mathbf{r_3} \rightarrow \left(v^1, v^2, v^3\right) = \left[\left(x_4 - x_3\right), \left(y_4 - y_3\right), \left(z_4 - z_3\right)\right]$$
$$\mathbf{w}^i = \mathbf{r_3} - \mathbf{r_2} \rightarrow \left(w^1, w^2, w^3\right) = \left[\left(x_3 - x_2\right), \left(y_3 - y_2\right), \left(z_3 - z_2\right)\right]$$

We calculate the cross product $\mathbf{p}_i$ of vectors $\mathbf{u}^i$ and $\mathbf{w}^i$ ($\mathbf{p}_i$ is a vector normal to the plane defined by the vectors $\mathbf{u}^i$ and $\mathbf{w}^i$) and the cross product $\mathbf{q}_i$ of the vectors $\mathbf{v}^i$ and $\mathbf{w}^i$ ($\mathbf{q}_i$ is a vector normal to the plane defined by the vectors $\mathbf{v}^i$ and $\mathbf{w}^i$). Furthermore, we calculate the vectors $\mathbf{p}^i$ and $\mathbf{q}^i$ and the magnitude of the vector $\mathbf{w}^i$.

$$\mathbf{p}_i = \mathbf{G}_{ij}\mathbf{u}^i \wedge \mathbf{w}^j \rightarrow V\left(p_1, p_2, p_3\right) = V\left[\left(u^2 w^3 - u^3 w^2\right), \left(u^3 w^1 - u^1 w^3\right), \left(u^1 w^2 - u^2 w^1\right)\right]$$
$$\mathbf{p}^i = \mathbf{G}^{ij}\mathbf{p}_j$$
$$p^1 = g^{11}V\left(u^2 w^3 - u^3 w^2\right) + g^{12}V\left(u^3 w^1 - u^1 w^3\right) + g^{13}V\left(u^1 w^2 - u^2 w^1\right)$$
$$p^2 = g^{21}V\left(u^2 w^3 - u^3 w^2\right) + g^{22}V\left(u^3 w^1 - u^1 w^3\right) + g^{23}V\left(u^1 w^2 - u^2 w^1\right)$$
$$p^3 = g^{31}V\left(u^2 w^3 - u^3 w^2\right) + g^{32}V\left(u^3 w^1 - u^1 w^3\right) + g^{33}V\left(u^1 w^2 - u^2 w^1\right)$$
$$\left|p\right|^2 = \mathbf{p}^i \bullet \mathbf{p}_j$$
$$p = \sqrt{\left|p\right|^2}$$

$$\mathbf{q}_i = \mathbf{G}_{ij}\mathbf{v}^i \wedge \mathbf{w}^j \rightarrow V\left(q_1, q_2, q_3\right) = V\left[\left(v^2 w^3 - v^3 w^2\right), \left(v^3 w^1 - v^1 w^3\right), \left(v^1 w^2 - v^2 w^1\right)\right]$$
$$\mathbf{q}^i = \mathbf{G}^{ij}\mathbf{q}_j$$
$$q^1 = g^{11}V\left(v^2 w^3 - v^3 w^2\right) + g^{12}V\left(v^3 w^1 - v^1 w^3\right) + g^{13}V\left(v^1 w^2 - v^2 w^1\right)$$
$$q^2 = g^{21}V\left(v^2 w^3 - v^3 w^2\right) + g^{22}V\left(v^3 w^1 - v^1 w^3\right) + g^{23}V\left(v^1 w^2 - v^2 w^1\right)$$
$$q^3 = g^{31}V\left(v^2 w^3 - v^3 w^2\right) + g^{32}V\left(v^3 w^1 - v^1 w^3\right) + g^{33}V\left(v^1 w^2 - v^2 w^1\right)$$
$$\left|q\right|^2 = \mathbf{q}^i \bullet \mathbf{q}_j$$
$$q = \sqrt{\left|q\right|^2}$$

$$\left|w\right|^2 = \mathbf{w}^i \bullet \mathbf{G}_{ij}\mathbf{w}^j$$
$$w = \sqrt{\left|w\right|^2}$$

We have the following relationships, the scalar product:
$$\mathbf{p}^i \bullet \mathbf{q}_j = \left|\mathbf{p}\right|\left|\mathbf{q}\right|\cos\boldsymbol{\tau}$$
$$\cos\boldsymbol{\tau} = \frac{\mathbf{p}^i \bullet \mathbf{q}_j}{\left|\mathbf{p}\right|\left|\mathbf{q}\right|} = \frac{\left(\mathbf{p}^i \bullet \mathbf{q}_j\right)w}{pqw}$$
and the cross and triple products:
$$\mathbf{p}^i \wedge \mathbf{q}^j = \left|\mathbf{p}\right|\left|\mathbf{q}\right|\sin\boldsymbol{\tau}$$
$$\left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right) = \left|\mathbf{p}\right|\left|\mathbf{q}\right|\left|\mathbf{w}\right|\sin\boldsymbol{\tau}$$
$$\sin\boldsymbol{\tau} = \frac{\left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right)}{pqw}$$
Then we can calculate the tan $\boldsymbol{\tau}$:

$$\tan\boldsymbol{\tau} = \frac{\sin\boldsymbol{\tau}}{\cos\boldsymbol{\tau}} = \frac{\dfrac{\left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right)}{pqw}}{\dfrac{\left(\mathbf{p}^i \bullet \mathbf{q}_j\right)w}{pqw}} = \frac{\left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right)}{pqw} \cdot \frac{pqw}{\left(\mathbf{p}^i \bullet \mathbf{q}_j\right)w} = \frac{\left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right)}{\left(\mathbf{p}^i \bullet \mathbf{q}_j\right)w} = \frac{\mathbf{S}}{\mathbf{T}}$$

and obtain the angle $\boldsymbol{\tau}$ **:**

$$\boldsymbol{\tau} = \arctan\boldsymbol{\tau} = \arctan\left(\frac{\mathbf{S}}{\mathbf{T}}\right)$$

The differential equation has the following form:

$$\mathrm{d}(\boldsymbol{\tau}) = \mathrm{d}\left(\arctan\left(\frac{\mathbf{S}}{\mathbf{T}}\right)\right) = \frac{1}{1+\left(\dfrac{\mathbf{S}}{\mathbf{T}}\right)^2}\mathrm{d}\left(\frac{\mathbf{S}}{\mathbf{T}}\right) = \frac{\mathbf{T^2}}{\mathbf{T^2}+\mathbf{S^2}}\mathrm{d}\left(\frac{\mathbf{S}}{\mathbf{T}}\right) = \frac{\mathbf{T^2}}{\mathbf{T^2}+\mathbf{S^2}}\left(\frac{\mathbf{1}}{\mathbf{T}}\mathrm{d}\,S - \frac{\mathbf{S}}{\mathbf{T^2}}\mathrm{d}\,T\right) =$$

$$= \frac{\mathbf{1}}{\mathbf{T^2}+\mathbf{S^2}}\left(\mathbf{T}.\mathrm{d}\,\mathbf{S} - \mathbf{S}.\mathrm{d}\,\mathbf{T}\right)$$

Using the chain rule, we write the derivative of $\boldsymbol{\tau}$ with respect to the atomic coordinates *x,y,z* of the four atoms involved.

In the above equation, we have to calculate the derivatives $\frac{\partial \mathbf{S}}{\partial r_j}$ e $\frac{\partial \mathbf{T}}{\partial r_j}$ (r_j , j=1,4) to obtain $\frac{\partial(\boldsymbol{\tau})}{\partial r_j}$.

Rewrite **S** e **T**:

$$\mathbf{S} = \left(\mathbf{p}^i \wedge \mathbf{q}^j \bullet \mathbf{w}^j\right) = V\begin{vmatrix} p^1 & p^2 & p^3 \\ q^1 & q^2 & q^3 \\ w^1 & w^2 & w^3 \end{vmatrix} = V\begin{vmatrix} \mathbf{G}^{ij}\mathbf{p}_j \\ \mathbf{G}^{ij}\mathbf{q}_j \\ \mathbf{w}^i \end{vmatrix} = V\mathbf{S'}$$

$$\mathbf{T} = \left(\mathbf{p}^i \bullet \mathbf{q}_j\right)w = \left(p^1q_1 + p^2q_2 + p^3q_3\right)w = \left(\mathbf{G}^{ij}\mathbf{p}_j \bullet \mathbf{q}_j\right)w = \mathbf{T'}\,w$$

where:

$$\mathbf{S'} = \begin{vmatrix} \mathbf{G}^{ij}V\left[\left(u^2w^3 - u^3w^2\right), \left(u^3w^1 - u^1w^3\right), \left(u^1w^2 - u^2w^1\right)\right] \\ \mathbf{G}^{ij}V\left[\left(v^2w^3 - v^3w^2\right), \left(v^3w^1 - v^1w^3\right), \left(v^1w^2 - v^2w^1\right)\right] \\ \left[w^1, w^2, w^3\right] \end{vmatrix} =$$

$$= \begin{vmatrix} \mathbf{G}^{ij}V\{[(y_1-y_2)(z_3-z_2)-(z_1-z_2)(y_3-y_2)], [(z_1-z_2)(x_3-x_2)-(x_1-x_2)(z_3-z_2)], [(x_1-x_2)(y_3-y_2)-(y_1-y_2)(x_3-x_2)]\} \\ \mathbf{G}^{ij}V\{[(y_4-y_3)(z_3-z_2)-(z_4-z_3)(y_3-y_2)], [(z_4-z_3)(x_3-x_2)-(x_4-x_3)(z_3-z_2)], [(x_4-x_3)(y_3-y_2)-(y_4-y_3)(x_3-x_2)]\} \\ \{(x_3-x_2), (y_3-y_2)(z_3-z_2)\} \end{vmatrix}$$

and

$$\mathbf{T'} = \mathbf{G}^{ij}V\{[(y_1-y_2)(z_3-z_2)-(z_1-z_2)(y_3-y_2)], [(z_1-z_2)(x_3-x_2)-(x_1-x_2)(z_3-z_2)], [(x_1-x_2)(y_3-y_2)-(y_1-y_2)(x_3-x_2)]\} \bullet$$
$$V\{[(y_4-y_3)(z_3-z_2)-(z_4-z_3)(y_3-y_2)], [(z_4-z_3)(x_3-x_2)-(x_4-x_3)(z_3-z_2)], [(x_4-x_3)(y_3-y_2)-(y_4-y_3)(x_3-x_2)]\}$$

For the vector $\mathbf{p}^i$, we have:

$$\frac{\partial \mathbf{p}^i}{\partial r} = \frac{\partial\left(\mathbf{G}^{ij}\mathbf{p}_j\right)}{\partial r} = \mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial r} + \mathbf{p}_j\frac{\partial \mathbf{G}^{ij}}{\partial r} = \mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial r}$$

being null the term $\mathbf{p}_j\frac{\partial \mathbf{G}^{ij}}{\partial r}$, and likewise for $\mathbf{q}^i$ **:**

$$\frac{\partial \mathbf{q}^i}{\partial r} = \mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial r}$$

Then we have to calculate:

$$\frac{\partial \mathbf{S}}{\partial r} = V\frac{\partial \mathbf{S'}}{\partial r} = V\left(\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial r} + \frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial r} + \frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\frac{\partial \mathbf{w}^i}{\partial r}\right)$$

and

$$\frac{\partial \mathbf{T}}{\partial r} = w\frac{\partial \mathbf{T}'}{\partial r} = w\left(\frac{\partial \mathbf{T}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial r} + \frac{\partial \mathbf{T}'}{\partial \mathbf{q}_i}\frac{\partial \mathbf{q}_i}{\partial r}\right)$$

The derivatives of **S'** e **T'** with respect to $\mathbf{r_1}$ (x_1,y_1,z_1) , $\mathbf{r_2}$ (x_2,y_2,z_2) , $\mathbf{r_3}$ (x_3,y_3,z_3) , $\mathbf{r_4}$ (x_4,y_4,z_4) are the following.

Atom 1

Parameter x_1

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_1} = \begin{vmatrix} \mathbf{G}^{ij}V\{0,-(z_3-z_2),(y_3-y_2)\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial x_1} = 0$$

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial x_1} = 0$$

$$\frac{\partial \mathbf{T}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_1} = \mathbf{G}^{ij}V\{0,-(z_3-z_2),(y_3-y_2)\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T}'}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial x_1} = 0$$

Parameter y_1

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_1} = \begin{vmatrix} \mathbf{G}^{ij}V\{(z_3-z_2),0,-(x_3-x_2)\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial y_1} = 0$$

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial y_1} = 0$$

$$\frac{\partial \mathbf{T}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_1} = \mathbf{G}^{ij}V\{(z_3-z_2),0,-(x_3-x_2)\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T}'}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial y_1} = 0$$

Parameter z_1

$$\frac{\partial \mathbf{S}'}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_1} = \begin{vmatrix} \mathbf{G}^{ij}V\{-(y_3-y_2),(x_3-x_2),0\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial z_1}=0$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial z_1}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_1}=\mathbf{G}^{ij}V\{-(y_3-y_2),(x_3-x_2),0\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial z_1}=0$$

Atom 2

Parameter x_2

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_2}=\begin{vmatrix}\mathbf{G}^{ij}V\{0,[(z_3-z_2)-(z_1-z_2)],[-(y_3-y_2)+(y_1-y_2)]\}\\ \mathbf{q}^i\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial x_2}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{0,-(z_4-z_3),(y_4-y_3)\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial x_2}=\begin{vmatrix}p^1 & p^2 & p^3\\ q^1 & q^2 & q^3\\ -1 & 0 & 0\end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_2}=\mathbf{G}^{ij}V\{0,[(z_3-z_2)-(z_1-z_2)],[-(y_3-y_2)+(y_1-y_2)]\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial x_2}=\mathbf{p}^i\bullet V\{0,-(z_4-z_3),(y_4-y_3)\}$$

Parameter y_2

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_2}=\begin{vmatrix}\mathbf{G}^{ij}V\{[-(z_3-z_2)+(z_1-z_2)],0,[(x_3-x_2)-(x_1-x_2)]\}\\ \mathbf{q}^i\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial y_2}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{(z_4-z_3),0,-(x_4-x_3)\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial y_2}=\begin{vmatrix}p^1 & p^2 & p^3\\ q^1 & q^2 & q^3\\ 0 & -1 & 0\end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_2}=\mathbf{G}^{ij}V\{[-(z_3-z_2)+(z_1-z_2)],0,[(x_3-x_2)-(x_1-x_2)]\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial y_2}=\mathbf{p}^i\bullet V\{(z_4-z_3),0,-(x_4-x_3)\}$$

Parameter z_2

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_2} = \begin{vmatrix} \mathbf{G}^{ij}V\{[(y_3-y_2)-(y_1-y_2)],[-(x_3-x_2)+(x_1-x_2)],0\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial z_2} = \begin{vmatrix} \mathbf{p}^i \\ \mathbf{G}^{ij}V\{-(y_4-y_3),(x_4-x_3),0\} \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial z_2} = \begin{vmatrix} p^1 & p^2 & p^3 \\ q^1 & q^2 & q^3 \\ 0 & 0 & -1 \end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_2} = \mathbf{G}^{ij}V\{[(y_3-y_2)-(y_1-y_2)],[-(x_3-x_2)+(x_1-x_2)],0\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial z_2} = \mathbf{p}^i\bullet V\{-(y_4-y_3),(x_4-x_3),0\}$$

<u>Atom 3</u>

Parameter x_3

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_3} = \begin{vmatrix} \mathbf{G}^{ij}V\{0,(z_1-z_2),[-(y_1-y_2)]\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial x_3} = \begin{vmatrix} \mathbf{p}^i \\ \mathbf{G}^{ij}V\{0,[(z_4-z_3)+(z_3-z_2)],[-(y_3-y_2)-(y_4-y_3)]\} \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial x_3} = \begin{vmatrix} p^1 & p^2 & p^3 \\ q^1 & q^2 & q^3 \\ 1 & 0 & 0 \end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_3} = \mathbf{G}^{ij}V\{0,(z_1-z_2),[-(y_1-y_2)]\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial x_3} = \mathbf{p}^i\bullet V\{0,[(z_4-z_3)+(z_3-z_2)],[-(y_3-y_2)-(y_4-y_3)]\}$$

Parameter y_3

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_3} = \begin{vmatrix} \mathbf{G}^{ij}V\{-(z_1-z_2),0,(x_1-x_2)\} \\ \mathbf{q}^i \\ \mathbf{w}^i \end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial y_3}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{[-(z_3-z_2)-(z_4-z_3)],0,[(x_4-x_3)+(x_3-x_2)]\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial y_3}=\begin{vmatrix}p^1 & p^2 & p^3\\ q^1 & q^2 & q^3\\ 0 & 1 & 0\end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_3}=\mathbf{G}^{ij}V\{-(z_1-z_2),0,(x_1-x_2)\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial y_3}=\mathbf{p}^i\bullet V\{[-(z_3-z_2)-(z_4-z_3)],0,[(x_4-x_3)+(x_3-x_2)]\}$$

Parameter z_3

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_3}=\begin{vmatrix}\mathbf{G}^{ij}V\{(y_1-y_2),-(x_1-x_2),0\}\\ \mathbf{q}^i\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial z_3}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{[(y_4-y_3)+(y_3-y_2)],[-(x_3-x_2)-(x_4-x_3)],0\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial z_3}=\begin{vmatrix}p^1 & p^2 & p^3\\ q^1 & q^2 & q^3\\ 0 & 0 & 1\end{vmatrix}$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_3}=\mathbf{G}^{ij}V\{(y_1-y_2),-(x_1-x_2),0\}\bullet\mathbf{q}_i$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial z_3}=\mathbf{p}^i\bullet V\{[(y_4-y_3)+(y_3-y_2)],[-(x_3-x_2)-(x_4-x_3)],0\}$$

<u>Atom 4</u>

Parameter x_4

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_4}=0$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial x_4}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{0,-(z_3-z_2),(y_3-y_2)\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial x_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial x_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial x_4}=\mathbf{p}^i\bullet V\{0,-(z_3-z_2),(y_3-y_2)\}$$

Parameter y_4

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_4}=0$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial y_4}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{(z_3-z_2),0,-(x_3-x_2)\}\\ \mathbf{w}^i\end{vmatrix}$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial y_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial y_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial y_4}=\mathbf{p}^i\bullet V\{(z_3-z_2),0,-(x_3-x_2)\}$$

Parameter z_4

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_4}=0$$

$$\frac{\partial \mathbf{S'}}{\partial \mathbf{q}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{q}_j}{\partial z_4}=\begin{vmatrix}\mathbf{p}^i\\ \mathbf{G}^{ij}V\{-(y_3-y_2),(x_3-x_2),0\}\\ \mathbf{w}^i\end{vmatrix}$$

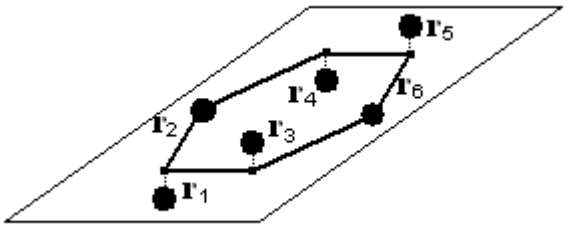
$$\frac{\partial \mathbf{S'}}{\partial \mathbf{w}^i}\cdot\frac{\partial \mathbf{w}^i}{\partial z_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{p}^i}\mathbf{G}^{ij}\frac{\partial \mathbf{p}_j}{\partial z_4}=0$$

$$\frac{\partial \mathbf{T'}}{\partial \mathbf{q}_i}\cdot\frac{\partial \mathbf{q}_i}{\partial z_4}=\mathbf{p}^i\bullet V\{-(y_3-y_2),(x_3-x_2),0\}$$

XIII. Planarity restraint

We define the least-squares plane the best plane through a group of atoms A, B, C,….The positions of the atoms (A, B,C,…) are represented by $\mathbf{r_n}=(x_n,y_n,z_n)$ (n = 1,2,3,…) and we want minimize the sum of the distances to the best plane.



$\mathbf{r_1} = (x_1, y_1, z_1)$
$\mathbf{r_2} = (x_2, y_2, z_2)$
$\mathbf{r}_3 = (x_3, y_3, z_3)$
.....
$\mathbf{r}_n = (x_n, y_n, z_n)$
The equation of the best fitting plane is:
$ax + by + cz + d = 0$
The distances of the atoms by the plane are:
$ax_1 + by_1 + cz_1 + d = d_1$
$ax_2 + by_2 + cz_2 + d = d_2$
$ax_3 + by_3 + cz_3 + d = d_3$
.....
$ax_n + by_n + cz_n + d = d_n$
We calculate the derivative of d_j with respect to r_j .
$$\frac{\partial d_j}{\partial r_j} = \frac{\partial (ax_j + by_j + cz_j + d)}{\partial r_j}$$

The derivatives with respect to the coordinates *x,y,z* of the atoms are:
Atomo j
$$\frac{\partial d_j}{\partial x_j} = a$$
$$\frac{\partial d_j}{\partial y_j} = b$$
$$\frac{\partial d_j}{\partial z_j} = c$$

XIV. Sum restraint

We want to keep constant the sum of the values of a parameter of some atoms. For example, in the space groups with floating origin along z-axis we keep constant the sum of the *z* coordinates of all the atoms.
$z_1 + z_2 + z_3 + ... + z_n = K$

The derivative of *K* with respect to z_j is:

$$\frac{\partial K}{\partial z_j} = \frac{\partial(z_1 + z_2 + z_3 + ... + z_n)}{\partial z_j} = 1$$

XV. Excessive shifts restraint

We want to impose limits against excessive shifts increasing the diagonal element of the normal matrix for the atomic parameter *p*. If σ is the maximum permissible breadth for the shift from the current value for this parameter, we can write:

$$M = \frac{(p - p_0)^2}{\sigma^2}$$

where p_0 is the former value of the parameter. The derivative of *M* with respect to *p* is:

$$\frac{\partial M}{\partial p} = \frac{1}{\sigma^2}$$

XVI. References

Andreev, Y.G., Lightfoot, P. & Bruce, P.G. *J. Appl. Cryst.* (1997), **30**, 294-305.

Burla, M.C., Caliandro, R., Camalli, M., Carrozzini, B., Cascarano, G.L., De Caro, L., Giacovazzo, C., Polidori, G. & Spagna, R. *J. Appl. Cryst.* (2005), **38**, 381-388.

Cruickshank, D.W. J. (1965), *Computing Methods in Crystallography*, pp. 112-116. Oxford: Pergamon Press.

Flack, H. D. *Acta Cryst*. (1983), **A39**, 876-881.

Giacovazzo, C. (2002), *Fundamentals of Crystallography*, 2nd edn. Oxford: Oxford University Press.

Herbest-Irmer, R. & Sheldrick, G. M. *Acta Cryst*.(1998), **B54**, 443-449.

Larson, A.C. (1969) *Crystallographic Computing,* pp. 291-294. Copenhagen: Munksgaard.

Schwarzenbach, D., Abrahams, S. C., Flack, H.D., Gonshorek, W., Hahn, T., Huml, K., Marsh, R., E., Prince, E., Robertson, B. E., Rollett, J.S. & Wilson, A.J.C. *Acta Cryst*. (1989), **A45**, 63-75.

Sheldrick, G.M. (1997), *SHELXL97. Program for the Refinement of Crystal Structures.* University of Goettingen, Germany.

Spagna, R. & Camalli, M. *J. Appl. Cryst.* (1999), **32**, 934-942.

Yeates, T. O. *Methods in Enzymology* (1997), **276**, 344-358.

Call for Contributions to the Next CompComm Newsletter

The eigth issue of the Compcomm Newsletter is expected to appear around November of 2007 with the primary theme to be determined. If no-one is else is co-opted, the newsletter will be edited by Lachlan Cranswick.

Contributions would be also greatly appreciated on matters of general interest to the crystallographic computing community, e.g. meeting reports, future meetings, developments in software, algorithms, coding, historical articles, programming languages, techniques and other news.

Please send articles and suggestions directly to the editor.

Lachlan M. D. Cranswick
Canadian Neutron Beam Centre (CNBC),
National Research Council of Canada (NRC),
Building 459, Station 18, Chalk River Laboratories,
Chalk River, Ontario, Canada, K0J 1J0
Tel: (613) 584-8811 ext: 3719
Fax: (613) 584-4040
E-mail: lachlan.cranswick@nrc.gc.ca
WWW: http://neutron.nrc-cnrc.gc.ca/peep_e.html#cranswick